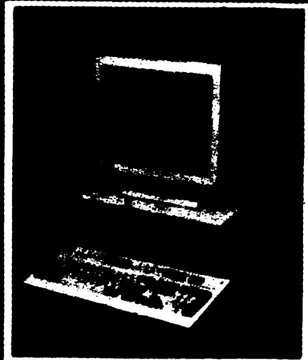
LUMINITA STATE



HELLO BASIC

Editura Agni

Luminiţa State

Hello, BASIC

Aplicaţii pe calculatoare de tip SPECTRUM

Ediţia a II - a

Editura Agni

Bucureşti 1993

Redactor: **Anton Bătătorescu**

Coperta: **Ioan Huţu**

Tehnoredactare computerizată: **REISOFT** tel: 6182431

Producător asociat: **D.P.P.T. - Bucureşti**

În aceeaşi serie vor apărea:

♦

Ion Diamandi - Calculatorul, coleg de bancă

(Matematică de gimnaziu pe calculator)

♦

Adrian Atanasiu - Algoritmul ? nimic mai simplu

(Exemplificări în BASIC şi PASCAL)

♦

Marian Gheorghe - Cine eşti tu, BASIC ?

(Iniţiere în limbajul de programare BASIC)

Cărţile publicate de editura noastră pot fi comandate pe adresa:

Editura AGNI CP: 30-107 Bucureşti

ISBN 973-95626-6-3

Copiilor mei,
Monica și Radu

CUVÎNTUL AUTORULUI

Limbajul BASIC, în diferitele lui variante, este unul dintre cele mai frecvent utilizate limbaje de programare. Ușor accesibil, chiar și persoanelor fără o instruire prealabilă în domeniul informaticii, limbajul BASIC poate fi rapid asimilat și oferă o modalitate comodă pentru reprezentarea și comunicarea de algoritmi și în același timp, suficient de puternică pentru rezolvarea unei game variate de probleme. Structurile limbajului BASIC, deși simple, comparativ cu alte limbaje de programare, permit totuși scrierea unor programe "disciplinate", facilitate de grafică, cromatică și sonoristică. Aceste posibilități prezente pentru majoritatea tipurilor de microcalculatoare, fac posibile vizualizări "atractive" pentru rezultatele - soluții ale problemelor rezolvate cu calculatorul.

Lucrarea "Hello, BASIC" își are sursa în activitatea de laborator de informatică desfășurată cu studenții anului I al Facultății de Matematică, pe microcalculatoare SPECTRUM, HC, TIM-S, LASER. Numeroasele teste vizînd dezvăluirea "anatomiei" limbajului, au pus în evidență o serie de particularități interesante care ne-au convins că deși "simplu" comparativ cu "frații lui mai mari", PASCAL, FORTRAN, limbajul BASIC este adecvat rezolvării multor probleme, în special a acelor probleme în care calculele matematice sînt predominante.

Prima parte a cărții realizează prezentarea versiunii BASIC implementată pe calculatoare compartibile SPECTRUM. Sperăm că numeroasele exemple care însoțesc prezentarea vor contribui substanțial la înțelegerea structurilor de bază și a modalităților de utilizare combinată a acestora în cadrul unui program.

Partea a doua a cărții este o colecție de programe organizată pe secțiuni. Programele incluse în cadrul colecției sînt direct utilizabile și se referă în principal la rezolvarea unor probleme de interes științific cu frecvență mare în cadrul aplicațiilor: calcul matriceal, prelucrări statistice, programare liniară. Posibilitățile de grafică și animație oferite de BASIC sînt ilustrate pe programele de tip **joc**, prezentate în cadrul capitolului **15**.

Aprecierea gradului de dificultate corespunzător diferitelor aplicații este formulată prin marcajul asociat programului astfel: marcajul "*"indică probleme relativ șimple, problemele marcate "**" le apreciem ca fiind de dificultate medie, problemele marcate "***" fiind în opinia noastră mai dificile fie din punctul de vedere al conținutului, fie ca efort de programare. Cu excepția programelor în care elementele de grafică, cromatică, animație intervin în mod esențial, programele prezentate pot fi adaptate, eventual cu lejere modificări pentru a putea fi utilizate pe mediile BASIC existente pe calculatoarele PC compatibile IBM.

Modul în care lucrarea a fost alcătuită corespunde intenției de a fi accesibilă și utilă unei categorii largi de cititori: elevii claselor de liceu, profesorii implicați în predarea noțiunilor de informatică în învățămîntul preuniversitar, studenții institutelor cu profil tehnic, economic, precum și persoanelor interesate în utilizarea calculatorului ca auxiliar în desfășurarea unor activități practice sau științifice.

Mulțumim colegilor, și studenților care au participat la elaborarea unora dintre programele prezentate în cadrul lucrării și în special domnului Anton Bătătorescu, cercetător la CCUB, pentru efortul deosebit depus în sprijinirea apariției acestei cărți.

Luminița State

CUPRINS

I. Limbajul BASIC pentru calculatoare compatibile SPECTRUM

1. Despre limbajul BASIC

1.1 Scurt istoric

Limbajul BASIC este în prezent limbajul de programare cel mai frecvent utilizat în microinformatică datorită facilităţilor conversaţionale, cromatice şi grafice pe care le oferă. Utilizînd notaţiile matematice uzuale şi un număr relativ mic de cuvinte provenind din limba engleză, limbajul BASIC, deşi uşor de asimilat chiar şi de către persoane ce nu posedă cunoştinţe de informatică, oferă în acelaşi timp programatorilor avansaţi, posibilitatea utilizării unor tehnici speciale permiţînd rezolvarea unei game largi de probleme pe un microcalculator.

Desigur, performanţele unui program scris în limbajul BASIC sînt modeste în comparaţie cu eficienţa ce poate fi obţinută pentru rezolvarea aceleiaşi probleme utilizînd un limbaj de programare de nivel înalt (Pascal, Fortran–77 etc.), dezavantaje compensate parţial de uşurinţa cu care acest limbaj poate fi învăţat şi de faptul că pentru rezolvarea unei probleme este suficient ca echipament de calcul un microcalculator.

Limbajul BASIC a fost creat în jurul anului 1965 la colegiul din Dartmouth, autorii primei versiuni fiind J.C.Nemeny şi Th.E.Kurtz. Numele limbajului s-a obţinut, ca în cazul majorităţii numelor limbajelor de programare, prin abrevierea unui nume mai lung ce indică şi intenţia pe care au avut-o autorii, anume, acesta fiind **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. Evoluţia limbajului BASIC în scopul ameliorării în special a flexibilităţii în introducerea datelor, precum şi în obţinerea unor variante implementabile pe minicalculatoare şi microcalculatoare compatibile cu sisteme de operare, a condus la dezvoltarea mai multor versiuni dintre care cităm BASIC-11, BASIC PLUS, BASIC-PLUS-2, BETA-BASIC, BASIC-80 etc.

În cadrul acestei lucrări vom prezenta varianta limbajului BASIC utilizată pentru microcalculatoarele de tip TIM-S, HC-85, ZX-SPECTRUM etc.

Toate exemplele ilustrative din carte au fost testate pe microcalculatoare de tip TIM-S şi ZX-SPECTRUM.

1.2 Ce este un program BASIC

O **instrucţiune** sau o **comandă BASIC** este o structură conformă cu anumite reguli de sintaxă ce reprezintă codificarea (în limbajul BASIC) unei acţiuni pe care utilizatorul doreşte să o întreprindă. In cazul microcalculatoarelor, o instrucţiune (comandă) indicată poate fi executată imediat sau poate fi introdusă ca o componentă a unei secvenţe de instrucţiuni ce descrie transcrierea în limbajul BASIC a acţiunilor necesare pentru rezolvarea unei anumite probleme, secvenţă ce este numită **program**.

Un **program BASIC** este o secvenţă de linii de program, fiecare linie de program constînd dintr-una sau mai multe instrucţiuni. Numărul maxim de instrucţiuni ce pot fi grupate pe aceeaşi linie de program depinde de caracteristicile microcalculatorului (minicalculatorului) particular pe care se operează. Fiecare linie de program are asociat un număr de ordine la care ne vom referi convenţional şi prin termenul de **etichetă**. Intr-un program BASIC numerele de ordine corespunzătoare liniilor de program sînt în ordine strict crescătoare, nefiind obligatoriu să fie numere consecutive.

Introducerea unei linii într-un program se realizează prin tastatura calculatorului, caracterele fiind afişate în partea inferioară a ecranului pe măsură ce tastele claviaturii sînt acţionate de către utilizator. Introducerea efectivă a liniei în cadrul programului se realizează prin acţionarea tastei [ENTER] (sau [CR] la anumite calculatoare).

În cazul în care o linie de program conţine mai multe instrucţiuni, acestea sînt separate prin separatorul ":" . Fiecare linie de program are asociat de către utilizator un număr de ordine care precede informaţia propriu-zisă reprezentată de secvenţa de instrucţiuni. In momentul în care linia este introdusă (prin acţionarea tastei [ENTER]) linia va fi inserată în program astfel încît să se asigure ordinea strict crescătoare numerelor de ordine a liniilor de program componente.

De exemplu, presupunînd că au fost deja introduse în program liniile de program avînd numerele de ordine 10 şi respectiv 20.

```
10 LET a = 20
20 PRINT a;b
```

în urma tastării şi introducerii liniei de program

```
15 LET b = a+2
```

programul va fi afişat pe ecran astfel:

```
10 LET a = 20
15 LET b = a+2
20 PRINT a;b
```

În consecinţă, se recomandă folosirea de numere neconsecutive pentru liniile de program, ceea ce crează posibilitatea inserării ulterioare a unor noi linii de program.

1.3 Editarea unui program

Introducerea într-un program a unei linii de program a cărei etichetă coincide cu numărul de ordine al unei linii deja existente în program determină înlocuirea liniei vechi cu noua linie.

De exemplu, dacă liniile de program sînt

```
10 LET a = 20
20 PRINT a;b
```

şi se tastează (şi se introduce) linia de program `20 LET b=3`, programul va fi

```
10 LET a = 20
20 LET b = 3
```

In acest mod se poate realiza, de exemplu, corectarea unei linii de program şi anume prin tastarea încă odată a etichetei corespunzătoare liniei pe care dorim să o corectăm, urmată de noul conţinut. Deşi posibil, acest mod de efectuare a corecţiilor este incomod, deoarece presupune tastarea în întregime a informaţiei din linia respectivă de program.

Eliminarea unei linii de program poate fi realizată prin tastarea etichetei corespunzătoare liniei urmată de acţionarea tastei [ENTER] (sau [CR]).

Corectarea unei linii de program poate fi realizată mult mai comod în maniera descrisă în continuare. Calculatorul dispune de un sistem de control asupra liniilor de program. Acesta este vizualizat pe ecran prin simbolul '>' (numit **prompter**) poziţionat în dreptul ultimei linii de program care a fost introdusă. De exemplu, dacă liniile de program au fost introduse în ordinea 10, 20, 15, atunci pe ecran prompterul va fi poziţionat în dreptul liniei de program cu eticheta 15 astfel:

```
10 LET a = 3
15 > LET b=5
20 PRINT a;b
```

Prin acţionarea tastelor pe care sînt înscrise simbolurile [↑] şi [↑] se determină deplasarea prompterului către liniile de numere de ordine inferioare, respectiv superioare etichetei corespunzătoare liniei curente.

Prin acţionarea tastei [EDIT] (sau simultan tastele [CAPS SHIFT] şi [1]) se realizează copierea în porţiunea inferioară a ecranului a liniei de program indicată de către prompter.

Calculatorul dispune de un control asupra poziţiei curente pe ecran, unde poate fi continuată scrierea, vizualizat printr-un cursor ce indică, pe de o parte, poziţia curentă pe ecran şi, pe de altă parte, modul de lucru în care operează la momentul respectiv (modul de lucru fiind indicat prin prezenţa unuia dintre caracterele **K**, **L**, **C**, **E**, **G** înscris pe cursor).

Prin acţionarea tastelor [←] şi [→], se realizează deplasarea cursorului poziţie cu poziţie către stînga, respectiv către dreapta poziţiei curente. Deplasînd cursorul pe poziţia imediat la dreapta zonei din linia de program pe care urmărim să o corectăm şi apăsînd tasta [DELETE] (sau simultan tastele [CAPS SHIFT] şi [0]) se realizează ştergerea caracter cu caracter a zonei respective. Tastînd în continuare informaţia ce dorim să fie inclusă în această zonă se obţine corecţia dorită. Acţionarea tastei [ENTER] (sau [CR]) va determina introducerea în program a liniei astfel corectate. Evident, dacă este necesară corectarea unei linii de program ce nu a fost încă introdusă în program, atunci este suficientă utilizarea numai a tastelor [←], [→] şi eventual [DELETE] pentru efectuarea corecţiei dorite.

Listarea liniilor unui program creat se realizează prin comanda LIST sau LIST n, unde n este un număr natural (urmată de acţionarea tastei [ENTER]).

Efectul unei astfel de acţiuni este de a lista toate liniile programului (în cazul comenzii LIST), respectiv de a obţine listarea numai a liniilor programului de la linia de program de etichetă p, unde p este cel mai mic număr de linie de program cu $p \geq n$.

În limbajul BASIC, LIST (respectiv LIST n) poate fi utilizată şi drept comandă (independentă de program), dar poate figura şi ca instrucţiune într-un program.

1.4 Execuţia unui program

Lansarea în execuţie a unui program este realizată prin comanda RUN (sau RUN n unde n este un număr natural) urmată de acţionarea tastei [ENTER] (sau [CR]).

Comanda RUN determină declanşarea execuţiei programului respectiv, iar comanda RUN n determină începerea execuţiei instrucţiunilor din program cu prima instrucţiune aflată pe linia de program de etichetă p, unde p este cel mai mic număr de ordine corespunzător unei linii de program (cu $p \geq n$).

In limbajul BASIC, RUN respectiv (RUN n) poate fi utilizată şi ca instrucţiune program.

Executarea comenzii RUN (sau RUN n) determină dispariţia de pe ecran a liniilor programului (programul rămîne însă în memoria calculatorului şi poate fi eventual relansat în execuţie pînă la o nouă resetare a calculatorului sau pînă la executarea unei comenzi NEW). Suspendarea execuţiei unui program poate fi realizată prin acţionarea tastei [BREAK] ceea ce determină afişarea unui mesaj în care este indicat şi numărul liniei program ce conţine instrucţiunea aflată în execuţie în momentul la care a survenit întreruperea. Determinarea reluării execuţiei instrucţiunilor din program se realizează prin comanda CONTINUE (acţionînd tasta [CONTINUE]), urmată de acţionarea tastei [ENTER] (sau [CR]). Comanda CONTINUE determină reluarea execuţiei programului începînd cu instrucţiunea imediat următoare în program celei aflate în curs de execuţie în momentul în care a survenit întreruperea comandată prin BREAK. Comenzile BREAK şi CONTINUE pot fi date şi în timpul efectuării transferului de informaţie între casetofon, dischetă, respectiv imprimantă şi memoria calculatorului. Afişarea

din nou a liniilor programului se realizează indicînd comanda LIST.

Instrucțiunile unui program se execută secvențial. Dacă o linie de program constă din mai multe instrucțiuni, acestea se execută secvențial în ordinea de la stînga la dreapta. Există însă și posibilitatea întreruperii ordinii secvențiale prin utilizarea instrucțiunilor de salt (GO TO, GO SUB), care determină ca execuția să nu mai fie realizată parcurgînd pe rînd fiecare linie de program, ci realizează trecerea la linia de program indicată și continuarea prin executarea instrucțiunilor componente ale acesteia. Oprirea execuției unui program se obține automat cînd a fost executată ultima instrucțiune de pe ultima linie de program (sau prin executarea unei instrucțiuni STOP (sau END), dacă există în program).

La un moment dat, în memoria calculatorului nu poate exista decît un singur program; toate liniile introduse sînt considerate ca făcînd parte din același program. Dacă se dorește reluarea programului respectiv, după o eventuală resetare, atunci el va trebui să fie introdus încă o dată prin tastatură (ceea ce este foarte incomod) sau poate fi copiat de pe bandă magnetică sau de pe dischetă. Ștergerea din memorie a unui program se poate realiza bineînțeles printr-o resetare (ceea ce nu recomandăm!) sau prin indicarea comenzii NEW (urmată de acționarea tastei ENTER).

Ca și comenzile LIST, RUN, comanda NEW poate să fie utilizată și ca instrucțiune într-un program.

In continuare, vom prezenta instrucțiunile BASIC ce permit scrierea de programe în care nu intervin elemente de grafică și nici nu se intenționează utilizarea posibilităților cromatice și "muzicale" ale microcalculatorului. Instrucțiunile care permit exploatarea acestor facilități vor fi prezentate în cadrul paragrafelor următoare. Toate instrucțiunile (comenzile) BASIC sînt identificate de cîte un cuvînt cheie, fiecare cuvînt cheie fiind considerat ca fiind un caracter și avînd asociat un cod ASCII (de exemplu, codul ASCII pentru cuvîntul cheie PRINT este 245).

Majoritatea instrucțiunilor BASIC pot fi utilizate și drept comenzi. In cazul utilizării ca o comandă, o instrucțiune BASIC este dată calculatorului fără etichetă, ea fiind executată imediat ce este introdusă în calculator (prin acționarea tastei ENTER). Pentru ca o instrucțiune să fie considerată ca o componentă a unui program, ea trebuie introdusă în calculator într-o linie de program (căreia îi este asociată o etichetă). Executarea instrucțiunilor introduse ca instrucțiuni ale unui program nu este realizată imediat după ce este introdusă în cadrul programului, ci numai după ce programul a fost lansat în execuție (cu comanda RUN sau RUN n).

1.5 Despre ecran

Accesarea ecranului se poate realiza în două maniere de lucru:

- în modul de lucru cu putere rezolutivă mică
- în modul de lucru cu putere rezolutivă mare

Accesarea ecranului în **modul de lucru cu putere rezolutivă mare** se realizează în scopuri legate de grafică, (vezi paragraful destinat facilităţilor în grafică).

În cadrul acestui paragraf vom considera în exclusivitate **modul de lucru în rezoluţie de mică putere.**

În acest mod de lucru, ecranul este partajat în 24x32 celule, dispuse pe 24 linii şi 32 coloane, în fiecare celulă putînd fi înscris un singur caracter. Referirea la o anumită celulă a ecranului se realizează prin indicarea numerelor corespunzînd liniei şi respectiv coloanei celulei vizate.

1.6 Convenţii de reprezentare

Pentru descrierea structurilor sintactice ale entităţilor limbajului BASIC vom utiliza următoarele convenţii de notaţie:

[   ] - pentru desemnarea tastelor (cheilor) tastaturii (claviaturii) terminalului.

Exemplu. [ENTER] , [RUN]

< > - pentru specificarea tipului de entitate ce poate fi utilizat în contextul respectiv

Exemplu. <variabilă>, <cuvînt cheie>, <etichetă>...

{ } - pentru indicarea entităţilor a căror apariţie este opţională.

Exemplu. { <variabilă> }

::= - pentru definirea diferitelor entităţi

Exemplu. <comentariu> :: ={<etichetă>} REM <text>

| - pentru indicarea alternativelor posibile în cazul considerat (a se citi **sau**)

Exemplu. <cifră>:: = 0|1|2|3|4|5|6|7|8|9

.. - pentru indicarea prezenţei tuturor elementelor dintr-o mulţime pe care este definită convenţional o relaţie de ordine totală

Exemplu. <cifră>:: = 0|1|..|9

' - ' pentru indicarea prezenţei unui anumit simbol

( ) * - pentru specificarea posibilității de repetare ori de cîte ori (eventual niciodată) a entității respective.

Entitățile limbajului BASIC sînt:

<caracter>, <constantă numerică>, <constantă șir>, <variabilă numerică>, <variabilă șir>, <tablou de tip numeric>, <tablou de tip șir>, <variabilă indexată de tip șir>, <expresie>, <operator aritmetic>, <operator relațional>, <operator logic>, <cuvînt cheie>, <instrucțiune>, <comandă>, <etichetă>.

2. Expresii BASIC

2.1 Constante, variabile

Simbolurile acceptate de limbajul BASIC sînt literele alfabetului latin (caractere format mic și majuscule), cifrele bazei 10, caracterele speciale, caracterele grafice ale limbajului, caracterele grafice create de către utilizator, precum și o mulțime de simboluri pe care le vom numi **cuvinte cheie.**

Reprezentarea internă a caracterelor acceptate de limbajul BASIC este conformă codului ASCII (**A**merican **N**ational **S**tandard **C**ode for **I**nformation **I**nterchange), lista completă a setului de caractere BASIC împreună cu codurile ASCII corespunzătoare fiind prezentată în Anexa 1.

Utilizînd convențiile de notație introduse, obținem următoarele descrieri:

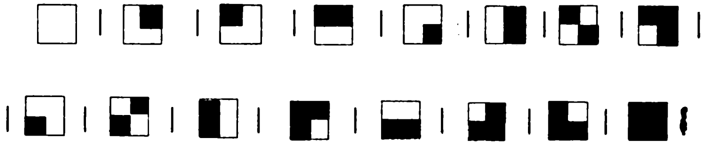
<caracter>:: = | <literă> | <cifră> | <caracter special> |
<cuvînt cheie> | <caracter grafic>

<literă>:: = A | B | .. | Z | a | b | .. | z

<cifră>:: = 0 | 1 | 2 | .. | 9

<caracter special>:: = + | - | * | = | / | " | > | < | > = | < > | (|) | $ | : | , | ; | < = | ↑ | @ |

<simbol grafic>:: =



<cuvînt cheie>:: = PRINT | EDIT | INK | .. (lista completă este prezentată în Anexa 1)

Prin simbolul "¤" convenim să reprezentăm un **blank** (spațiu liber).

Constante

Constantele limbajului BASIC sînt de două tipuri: **constante numerice** și **constante string** (șir).

Constantele numerice sînt de tip real și sînt memorate cu precizie de 9 sau 10 cifre. Numărul maxim ce poate fi reprezentant este de ordinul 10^{38}, numărul minim pozitiv (considerat ca fiind diferit de 0) este aproximativ 4.10^{-39}.

Reprezentarea internă a unei constante numerice este în virgulă mobilă, utilizînd un **byte** pentru reprezentarea exponentului e ($1<e<255$) și 4 bytes pentru reprezentarea mantisei m ($1/2 \leq m < 1$), reprezentarea corespunzînd numărului $m.2^{e-128}$.

Pentru scrierea unei constante numerice se acceptă cel mult 14 caractere. Deoarece $1/2 \leq m < 1$ rezultă că bitul cel mai semnificativ al mantisei este întotdeauna 1, din acest motiv acest bit este utilizat pentru reprezentarea semnului corespunzător numărului și anume, prin convenție:

0 corespunde semnului '+' și 1 corespunde semnului '–' .

Constantele numerice pot fi date ca informație calculatorului, fie în manieră uzuală, de exemplu 5, –17.25, 127.335, 0,5 (sau .5), fie printr-un astfel de număr împreună cu un exponent reprezentat prin litera e (sau E), urmat de un număr întreg cu cel mult două cifre. Această reprezentare corespunde numărului obținut prin înmulțirea numărului ce precede litera e (sau E) cu puterea lui 10 indicată de numărul întreg ce urmează literei e.

Exemplu. 2.34e2, – 2.34e+2, 2.34e–21

sînt reprezentări echivalente respectiv pentru constantele numerice

234, –234, $2.34.10^{-21}$.

Deoarece precizia este 9 (sau 10) cifre, rezultă că 1e10 este deja prea mare pentru ca să fie posibilă reprezentarea exactă a acestuia. Astfel, numerele 1e1 și 1e10 +1 vor fi considerate de către calculator ca fiind egale. O situație similară este prezentată de numerele 5e9+1 lu 5e9+2, care, de asemenea, vor fi considerate de către mașină ca fiind egale.

Constantele de tip string (șir)

O constantă de tip **string** este o secvență de caractere BASIC delimitată de simbolul ' " '.

Exemplu. "Aceasta este o constantă string".

Simbolurile delimitatoare pentru reprezentarea unei constante string nu sînt componente ale secvenței reprezentînd constanta respectivă. Secvența de caractere ce constituie o constantă string poate să nu conțină nici un caracter, caz în care constanta se reprezintă prin " " (stringul vid). Dacă printre caracterele care compun secvența ce corespunde unei constante string apare caracterul ' " ' atunci va trebui să fie reprezentată prin ' " " '.

Exemplu: "DON" "T".

Observaţie. In cazul constantelor de tip string, caracterele de tip literă format mic sînt considerate ca fiind distincte de majusculele corespunzătoare.

Exemplu: Constantele "a" şi "A" sînt considerate de către calculator ca fiind distincte.

Utilizînd notaţiile introduse obţinem pentru entitatea <constantă string> descrierea sintactică:

<constantă string> :: = "(caracter)*"

Variabile BASIC

In general, prin termenul de **variabilă** se înţelege o entitate caracterizată prin atributele: **identificator** (nume), **tip**, **adresă**, **valoare**. Din punct de vedere fizic identificatorul, reprezentînd numele variabilei, desemnează o zonă de memorie **contiguă**, primul byte avînd adresa corespunzătoare variabilei. In zona respectivă este eventual înregistrată o informaţie ce reprezintă valoarea variabilei. Pentru un anumit identificator utilizat ca nume al unei variabile calculatorul, asociază în mod automat o adresă (fără controlul utilizatorului), valoarea variabilei putînd fi modificată ca efect al executării unor anumite comenzi sau instrucţiuni. Referirea la valoarea unei variabile (informaţia conţinută în zona de memorie asociată) se realizează prin utilizarea identificatorului selectat ca nume al variabilei.

Variabilele acceptate de limbajul BASIC pot fi de unul din tipurile: **numeric** sau **string** (şir).

Variabilele de tip numeric (respectiv string) pot fi **variabile simple** sau **variabile indexate de tip numeric** (respectiv string).

Variabile simple de tip numeric

Identificatorul corespunzător unei variabile simple de tip numeric poate fi orice secvenţă de litere, şi/sau cifre. Primul caracter al secvenţei trebuie să fie o literă.

<identificator variabilă de tip numeric>:: = <literă>(<literă> | <cifra>)

Exemple. ab1c, Ab, variabila, A123

Valoarea unei variabile de tip numeric poate fi nedefinită (în zona de memorie asociată nu a fost înregistrată informaţie) sau poate fi o constantă de tip numeric.

Variabile simple de tip string

Identificatorul corespunzător unei variabile de tip string este o secvenţă compusă dintr-o literă urmată de caracterul '$'.

<identificator variabilă de tip string>:: = <literă>$.

Exemple. a$, t$, A$, C$

Valoarea unei variabile de tip string poate fi nedefinită sau poate fi orice constantă de tip şir (inclusiv stringul vid " ").

Observaţie. O caracteristică a majorităţii variantelor de BASIC pentru microcalculatoare este aceea că, în secvenţele identificatori de variabile, calculatorul nu face distincţie între literele format mic şi majusculele respective.

De exemplu, perechile de identificatori AbC, abc, respectiv A$. a$ vor fi considerate ca fiind asociate respectiv aceleeaşi variabile.

2.2 Sintaxa expresiilor aritmetico-logice

Expresiile în limbajul BASIC reprezintă structuri rezultate prin concatenarea variabilelor, constantelor, operatorilor aritmetici, operatorilor relaţionali, operatorilor logici utilizînd eventual parantezele (,), apelări de funcţii BASIC sau funcţii definite de utilizator, după modelul general al construcţiilor de tip expresie **algebrică** şi **logică**. Limbajul BASIC oferă însă şi posibilitatea unor construcţii pe care le vom numi tot **expresii**, care nu mai au corespondent în calculul algebric, sau calculul logic uzual. Din punct de vedere sintactic, conceptul de **expresie** poate fi definit în modul următor:

<expresie>:: =

<variabilă> | <variabilă indexată> | <apelare funcţie> | <constantă> | <expresie> <operator> <expresie> | (<expresie>) <operator> <expresie> | <expresie> <operator> (<expresie>) | (<expresie>) | -(<expresie>) | +(<expresie>)

<operator> :: = <operator aritmetic> | <operator relaţional> | <operator logic>

<operator aritmetic>:: = | + | - | * | ↑ | /

<operator relaţional>:: = < | < = | > | > = | < > | =

<operator logic>:: = AND | OR | NOT

Semnificaţiile **operatorilor aritmetici** sînt:

- \+ operaţia de adunare
- \- operaţia de scădere
- * operaţia de înmulţire
- / operaţia de împărţire
- ↑ operaţia de exponenţiere (cu baza strict pozitivă)

Semnificaţiile **operatorilor relaţionali** sînt:

- = egalitate
- < > diferit
- < = mai mic sau egal

>= mai mare sau egal

< mai mic

> mai mare

Semnificaţiile **operatorilor logici** sînt:

NOT negaţia logică

OR disjuncţia logică

AND conjuncţia logică

Descrierea sintactică dată pentru conceptul de expresie permite obţinerea şi a unor construcţii ilegale cum este de exemplu NOT"abc". Pentru eliminarea acestei posibilităţi este necesară impunerea unor condiţii suplimentare "de bună formare" pentru structurile acceptate de limbajul BASIC ca expresii. Pentru simplificarea prezentării preferăm să precizăm care construcţii, în afara construcţiilor de tip expresie aritmetică, sau expresie logică, sînt admise în cadrul limbajului BASIC.

2.3 Evaluarea expresiilor

Evaluarea unei expresii în care nu intervin decît variabile, constante de tip numeric, apelări de funcţii ce conduc la rezultate de tip numeric şi operatori aritmetici se realizează după regulile de calcul algebric şi a priorităţilor uzuale: evaluarea subexpresiilor incluse între paranteze, efectuarea apelărilor funcţiilor etc. Pentru identificatorii de variabile ce apar în astfel de construcţii se consideră valorile curente asociate respectiv variabilelor.

Nivelele de prioritate pentru operatorii aritmetici sînt:

↑ prioritate maximă,

* şi / pe acelaşi nivel de prioritate,

\+ şi − pe ultimul nivel de prioritate.

Trebuie reţinut faptul că toţi operatorii aritmetici sînt asociativi stînga, ceea ce înseamnă că evaluarea unei expresii care conţine numai operatori de acelaşi nivel de prioritate se realizează de la stînga la dreapta.

Exemplu. 3 ↑ 2 ↑ 3 = 729, însemnînd (3 ↑ 2) ↑ 3 = 729.

După cum rezultă din documentaţia elaborată pentru calculatorul aMIC,[14], operatorul '↑' este în cazul variantei de BASIC respective, operator asociativ dreapta (cum este şi în cazul altor limbaje de programare, de exemplu FORTRAN).

Valorile logice **adevărat** şi **fals** sînt reprezentate în calculator prin valorile numerice 1, respectiv 0.

Rezultatul evaluării unei expresii constînd dintr-o constantă este reprezentat de

acea constantă: rezultatul evaluării unei expresii constînd dintr-o variabilă este valoarea curentă corespunzătoare variabilei respective.

Pentru evaluarea unei expresii de tipul

<expresie 1> <operator relațional> <expresie 2>

se evaluează <expresie 1> și <expresie 2> și dacă au rezultat constante de tip numeric, atunci rezultatul va fi 1 sau 0 după cum relația este adevărată, respectiv falsă.

Exemplu. Rezultatul evaluării expresiei 2<3 este 1; rezultatul evaluării expresiei a < >2 este 1 dacă valoarea variabilei a este diferită de 2 și este 0 în caz contrar.

Pentru evaluarea unor expresii de tipul

<expresie 1> <operator logic> <expresie 2>

se evaluează <expresie 1> și <expresie 2> și dacă prin evaluarea celor două expresii au rezultat constante numerice fie acestea a,b atunci

$$\text{NOT } a = \begin{cases} 0 \text{ dacă } a \neq 0 \\ 1 \text{ dacă } a = 0 \end{cases}$$

Construcțiile de tipul aNOTb sînt ilegale.

$$a \text{ AND } b = \begin{cases} a \text{ dacă } b \neq 0 \\ 0 \text{ dacă } b = 0 \end{cases}$$

$$a \text{ OR } b = \begin{cases} 1 \text{ dacă } b \neq 0 \\ a \text{ dacă } b = 0 \end{cases}$$

Se observă că în cazul în care <expresie 1>, <expresie 2> sînt expresii logice (în sensul obișnuit), atunci evaluarea expresiilor logice de tipul NOT a, a OR b, a AND b corespunde regulilor de calcul boolean cunoscute.

Observații.

1. Dacă <expresie 1>, <expresie 2> sînt expresii prin ale căror evaluări rezultă constante de tip string atunci

<expresie 1> + <expresie 2>

este o expresie BASIC, rezultatul evaluării acestei expresii fiind constanta obținută prin concatenarea celor două constante șir.

Exemplu. Rezultatul evaluării expresiei

"abc" + "1234" este constanta "abc1234"

Dacă variabilele a$, b$ au drept valori constantele "1" și respectiv "-2" atunci rezultatul evaluării expresiei a$ + b$ va fi constanta string "1-2".

2. Operatorul AND poate fi utilizat în construcție de tipul

<expresie 1> AND <expresie 2>

unde prin evaluarea expresiei <expresie 1> rezultă o constantă de tip șir c, iar

prin evaluarea expresiei <expresie 2> rezultă o constantă de tip numeric b. Evaluarea unei astfel de expresii se realizează după regula:

$$c \text{ AND } b = \begin{cases} c & \text{dacă } b \neq 0 \\ \text{""} & \text{dacă } b = 0 \end{cases}$$

Exemplu. Rezultatul evaluării expresiei "abc" AND 3 este "abc", iar rezultatul evaluării expresiei a AND b este constanta string valoare a variabilei a dacă valoarea variabilei b este diferită de 0 şi stringul vid " " în caz contrar.

3. Pentru evaluarea unei expresii de tipul

<expresie 1> <operator relaţional> <expresie 2>

unde <expresie 1>, <expresie 2> sînt expresii logice uzuale, se efectuează evaluarea expresiilor care apar ca operanzi, evaluare care conduce la obţinerea respectiv a valorilor 0 (fals), 1 (adevărat), valori ce sînt comparate în sensul operatorului relaţional considerat, rezultînd în final una din valorile 0, 1.

4. Limbajul BASIC permite obţinerea unor structuri de tip expresie în care operatorii relaţionali au ca operanzi expresii prin ale căror evaluări se obţin respectiv constante de tip string.

Compararea în sensul unui operator relaţional a două constante de tip string presupune compararea în scris lexicografic a codurilor ASCII corespunzătoare perechilor de caractere care se află pe poziţii de acelaşi rang (de la stînga la dreapta), respectiv în cele două secvenţe de caractere. Rezultatul evaluării este 1 sau 0 după cum relaţia este adevărată, respectiv falsă.

Obţinem astfel următoarele reguli de evaluare:

a$ = b$ are valoarea 1, dacă şi numai dacă caracterele aflate pe poziţii de acelaşi rang respectiv în constantele care reprezintă valorile celor două variabile au acelaşi cod ASCII (coincid).

Exemple.

"abc" = "abc" are valoarea 1

"abc" = "aBc" are valoarea 0

a$ < > b$ are valoarea 1, dacă şi numai dacă valoarea expresiei a$ = b$ este 0.

a$ < b$ are valoarea 1, dacă şi numai dacă vectorul obţinut prin considerarea codurilor ASCII corespunzătoare caracterelor care formează constanta string valoare a$ variabilei a precede în sens lexicografic vectorul obţinut în acelaşi mod corespunzător variabilei b$.

a$< =b$ are valoarea 1 dacă şi numai dacă cel puţin una din expresiile a$=b$, a$<b$ are valoarea 1.

a$>b$ are valoarea 1, dacă şi numai dacă a$< =b$ are valoarea 0.

a$> =b$ are valoarea 1, dacă şi numai dacă a$<b$ are valoarea 0.

Expresii	Rezultatul obţinut prin evaluare
(3+ (2<3)) <4	0
3+ (NOT(2<3))<4	1
NOT (2<3)<= ((2=3) AND (3<4))	1
(2<3)<((2=3) OR (3<4))	0
NOT (2<3) =((2=3) OR(3<4))	0
NOT (2>=3) =((2=3) OR (3<4))	1
NOT (2<3)<>((2<3) OR (3=4))	1
NOT (2<3) + 3	3
"AAa123"<"AAa22345"	1
"AAa123"<"AAa2"	1
"a"<"A"	0
a" = "A"	0
"AAa123"<=AAa1234"	1
("123" + "abcd")<"123abc"	0
"123"+"abcd" = "123abcd"	1

Într-o construcţie de tip expresie pot exista operatori aritmetici, operatori relaţionali, operatori logici, apelări de funcţii, anumite subexpresii fiind eventual incluse între paranteze. Evaluarea unei astfel de expresii se realizează conform cu următoarele reguli de prioritate:

1. Se evaluează subexpresiile din paranteze.

2. Se realizează apelările de funcţii şi înlocuirile în subexpresiile respective ale acestora cu valorile rezultate.

3. Se evaluează subexpresiile corespunzătoare operatorilor aritmetici. Reamintim că subnivelele de prioritate în clasa operatorilor aritmetici sînt:

↑

*, /

+, -

Operatorii aritmetici sînt asociativi stînga.

4. Se evaluează subexpresiile corespunzătoare operatorilor relaţionali. Operatorii relaţionali au acelaşi nivel de prioritate şi evaluarea se realizează de la stînga la dreapta.

5. Se evaluează subexpresiile corespunzătoare operatorilor logici. Operatorii logici sînt ierarhizaţi pe subnivelele de prioritate

NOT
AND
OR

Evaluarea operatorilor logici aflaţi pe acelaşi subnivel de prioritate se realizează de la stînga la dreapta.

3. Funcţii BASIC

Limbajul BASIC permite efectuarea directă a anumitor operaţii asupra datelor de tip numeric respectiv de tip string prin intermediul unor funcţii predefinite. De asemenea, este posibil ca utilizatorul să-şi definească propriile funcţii. In afară de funcţiile ce vor fi prezentate în cadrul acestui paragraf, limbajul BASIC dispune de o serie de funcţii ce permit efectuarea de operaţii asupra informaţiei înscrise pe ecran (vezi paragraful destinat facilităţilor de grafică).

3.1 Operaţii cu stringuri: LEN, CHR$, VAL, STR$, SLICE

LEN

Funcţia LEN necesită un singur argument care poate fi o expresie prin a cărei evaluare rezultă o constantă de tip string.

Rezultatul apelării funcţiei LEN este numărul ce exprimă lungimea constantei (numărul de caractere ce compun constanta).

Exemple. LEN"abc" este 3

LEN" " este 0

LEN("abc" + "def") este 6

LEN"abc" + LEN"1" este 4

CHR$

Funcţia CHR$ necesită un singur argument care poate fi o expresie prin a cărei evaluare rezultă o constantă numerică, fie aceasta **c**. Dacă **y** este valoarea obţinută prin rotunjirea lui **c** (înlocuirea lui **c** prin cel mai apropiat număr întreg) atunci rezultatul apelării funcţiei CHR$ pentru argumentul considerat este (dacă există!) constanta string constînd din caracterul care are codul ASCII egal cu **y**.

Exemple. CHR$ 1e2 este constanta "d"
CHR$ 65 este constanta "A"
CHR$ 98.75 este constanta "c"
CHR$ 98.25 este constanta "b"
CHR$ 98.5 este constanta "c"
CHR$ 195 este constanta string constînd din caracterul "NOT"

CHR$ 200 este caracterul > =

Utilizarea următoarelor apelări conduc respectiv la afişarea mesajelor de eroare ce vor fi indicate.

CHR$ 12 afişează '?', deoarece codul ASCII 12 corespunde caracterului cuvînt cheie corespunzător comenzii DELETE.

CHR$ 256 mesaj de eroare "Integer out of range"

CHR$ -1 mesaj de eroare "Integer out of range"

VAL

Funcţia VAL necesită ca argument o expresie prin a cărei evaluare rezultă o constantă de tip string. Prin apelarea funcţiei VAL pentru un argument de tip string dat, componentele constantei sînt considerate ca fiind secvenţa de caractere corespunzătoare unei expresii BASIC şi se realizează evaluarea acestei expresii. In cazul în care componentele argumentului funcţiei VAL nu corespund sintaxei unei expresii (deci evaluarea nu este posibilă), atunci calculatorul afişează un mesaj de eroare.

In particular, funcţia VAL permite conversia automată a unui număr ale cărui cifre sînt reprezentate ca secvenţă de caractere în constanta numerică corespunzătoare (reprezentată în memorie prin indicarea mantisei şi caracteristicii).

Exemple. VAL ("123") + 2 este 125
VAL ("2*3") este 6
VAL ("2" + "*3") este 6
VAL ("2*"+"3") este 6
VAL ("9*" + "–3") este –27
VAL ("–3*"+"–2") este 6

Dacă variabila **a** are ca valoare curentă numărul 3 atunci

VAL ("a + 2") este 5.

Următoarele apelări ale funcţiei VAL sînt ilegale şi conduc la afişarea unor mesaje de eroare.

VAL"abc" mesaj de eroare "2 Variabile not found", deoarece structura abc este considerată ca fiind identificatorul unei variabile a cărei valoare este nedefinită.

VAL ("9*" + "/3") mesaj de eroare "c Nonsense in BASIC", deoarece secvenţa 9*/3 nu corespunde sintaxei unei expresii BASIC .

Dacă valoarea variabilei **a** este nedefinită atunci apelările
VAL ("a" +"+2") şi VAL ("a+2") conduc la afişarea mesajului de eroare "2 Variabile not found", deoarece este imposibilă evaluarea expresiei a+2.

VAL ("a" + "2") poate determina mesajul de eroare "2 Variable not found", dacă expresia **a2** corespunde unui identificator de variabilă a cărei valoare este nedefinită.

VAL ("2" + "a") mesaj de eroare "C Nonsense in BASIC", deoarece secvenţa 2a nu corespunde sintaxei unei expresii BASIC .

Operaţia de conversie a unui număr dat ca argument în constanta string ale cărei componente sînt caracterele care corespund cifrelor numărului se poate realiza prin utilizarea funcţiei STR$.

STR$

**Funcţia STR$** necesită un singur argument care trebuie să fie o expresie prin a cărei evaluare se obţine o constantă numerică. Rezultatul apelării funcţiei STR$ este constanta string ale cărei componente sînt caracterele corespunzătoare cifrelor numărului respectiv.

Exemple.

```
STR$ (2.5*2.5)                  este constanta string "6.25"
STR$ 12 + "a"                   este "12a"
VAL (STR$ 12)+3                 este 15
VAL STR$ 12+3                   este 15
STR$ 12.75                      este "12.75"
STR$ (2*3)+ CHR$ 65             este "6A"
VAL (CHR$ 54 + STR$ 2)          este  62
STR$ (2* -3)                    este "-6"
STR$ (-2*3)                     este "-6"
CHR$ VAL (CHR$ 54 + STR$ 2)     este " >"
STR$ (1|3)                      este "0.33333333"
STR$ 9e15                       este "9e +15"
STR$ 9e-15                      este "9e-15"
```

SLICE

Pentru obţinerea unei subsecvenţe a unei secvenţe de caractere reprezentînd o constantă string, majoritatea variantelor de limbaje BASIC pentru microcalculatoare oferă o facilitate specială numită operaţia **TO** sau operaţia "**slice**" (adoptînd termenul utilizat în documentaţia în limba engleză pentru denumirea acestei funcţii).

Descrierea sintaxei acestei funcţii este:

<slice> :: = <expresie> ({<expresie 1>} TO {<expresie 2>})

unde <expresie> este o expresie BASIC prin a cărei evaluare rezultă o constantă de tip string. Parametrii <expresie 1> şi <expresie 2> sînt opţionali şi reprezintă expresii prin ale căror evaluări rezultă valori numerice. In cazul în care parametrul <expresie 1> lipseşte, calculatorul presupune pentru el valoarea 1; în cazul în care parametrul <expresie 2> lipseşte, calculatorul presupune pentru el valoarea LEN <expresie>. Fie n,p valorile rezultate prin evaluările expresiilor

<expresie 1> şi respectiv <expresie 2> şi cărora li s-a aplicat operaţia de rotunjire.

Rezultatul aplicării funcţiei **slice** argumentului <expresie> este constanta string avînd drept componente caracterele aflate pe poziţiile de ranguri **n..p** din constanta obţinută prin evaluarea argumentului <expresie>. Dacă n>p atunci rezultatul aplicării funcţiei slice este stringul vid " ".

Exemple.

Expresie	Rezultat
"abcdef" (2 TO 5)	"bcde"
"abcdef" (2 TO)	"bcdef"
"abcdef" (TO 5)	"abcde"
"abcdef" (TO)	"abcdef"
"abcdef" (3 TO 3)	"c"
"abcdefh" ((2+1) TO (2*3	"cdef"
"abcdef"" (5 TO 3)	" "
"abcdef" (8 TO 7)	" "
"abcdef" (1 TO 0)	" "
"abcdef (3 TO 0)	" "
("abc" + "def") (2 TO 5)	"bcde"
"abc" + "def" (2 TO)	"abcef"
"abcdefgh" (1.5 TO 3.7)	"bcd"
"abcdefgh" (1.4 TO 3.7)	"abcd"
"abcdefgh" (0.7 TO 2)	"ab"
"abcdefgh" (1 TO 7.4)	"abcdefg"
"abcdefgh" (1 TO 7.5)	"abcdefgh"
"abcdefgh" (1 TO 8.2)	"abcdefgh"

Utilizarea funcţiei **slice** pentru valori neconcordante pentru <expresie 1>, <expresie 2> şi LEN <expresie> conduce la afişarea unor mesaje de eroare.

Exemple.

"abcdef" (– 1 TO 2) mesaj de eroare "B Integer out of range"

"abcdef" (2 TO 7) mesaj de eroare "3 Subscript wrong", deoarece LEN "abcdef" = 6

Utilizări de tipul "abcdef" (0 TO 2), "abcdef" (0.2 TO 3), "abcdef" (1 TO 6.5) conduc, de asemenea, la afişarea mesajului de eroare "3 Subscript wrong", deoarece valorile pentru rangurile poziţiilor de început şi sfîrşit al subsecvenţei indicate nu concordă cu lungimea constantei căreia îi este aplicată funcţia **slice**, respectiv nu sînt mai mari sau egale decît 1.

Observaţie. Majoritatea versiunilor BASIC care au inclusă funcţia **slice** acceptă ca scriere alternativă pentru <expresie> (n TO n) scrierea <expresie> (n). De exemplu, "abcdef" (3 TO 3) ca şi "abcdef" (3) reprezintă constanta string "c".

3.2 Operaţii aritmetice: SGN, ABS, INT, SQR

SGN

Funcţia SGN necesită un singur argument care poate fi o expresie prin a cărei evaluare se obţine o constantă de tip numeric. Sintaxa funcţiei SGN este

<funcţia SGN> :: = SGN <expresie>

Dacă **x** este valoarea obţinută prin evaluarea expresiei indicate, atunci rezultatul apelării funcţiei este:

$$SGN\ x = \begin{cases} 1 \text{ dacă } x > 0 \\ 0 \text{ dacă } x = 0 \\ -1 \text{ dacă } x < 0 \end{cases}$$

Exemple. SGN (3 <5) este 1
SGN VAL ("2" + "−3") este −1
SGN 3<0 este 0

ABS

Funcţia ABS necesită un singur argument care poate fi orice expresie prin a cărei evaluare rezultă o constantă de tip numeric. Sintaxa funcţiei ABS este

<funcţia SGN> :: = ABS <expresie>

Dacă **x** este valoarea rezultată prin evaluarea expresiei indicate ca argument, atunci prin apelarea funcţiei ABS se obţine

$$ABS\ x = \begin{cases} x \text{ dacă } x > 0 \\ 0 \text{ dacă } x = 0 \\ -x \text{ dacă } x < 0 \end{cases}$$

INT

Funcţia INT necesită indicarea unui singur argument ce poate fi orice expresie prin a cărei evaluare se obţine o constantă de tip numeric.

<funcţia INT> :: = INT <expresie>

Dacă **x** este valoarea obţinută prin evaluarea expresiei date ca argument atunci rezultatul aplicării funcţiei INT este partea întreagă a numărului **x**.

Exemple. INT 3.9 este 3
INT −3.9 este −4
INT 3.2 este 3
INT −3.2 este −4

Observaţie. Apelarea funcţiei INT nu este echivalentă cu aplicarea operaţiei de rotunjire. Operaţia de rotunjire aplicată unei valori **x**, spre deosebire de INTx, asociază numărul întreg cel mai apropiat de **x**.

SQR

Funcţia SQR permite calculul unei aproximaţii a radicalului de ordinul doi pentru argumentul indicat. Sintaxa funcţiei SQR este

<funcţia SQR>:: = SQR <expresie>

unde <expresie) este o expresie prin a cărei evaluare rezultă o valoare numerică nenegativă.

Exemple. SQR 4 este 2
SQR 2 este 1.4142136
SQR 0 este 0

Utilizarea ilegală a funcţiei SQR conduce la obţinerea unor mesaje de eroare. De exemplu, utilizarea apelării SQR-4 va conduce la afişarea mesajului de eroare "A Invalid argument".

3.3 Funcţiile trigonometrice: SIN, COS, TAN, ASN, ACS, ATN

SIN, COS, TAN

Funcţiile SIN, COS, TAN calculează respectiv aproximaţii pentru valorile **sin x, cos x, tg x,** unde **x** este argumentul ce exprimă măsura unghiului exprimată în radiani.

Sintaxa utilizării acestor funcţii este

<funcţia SIN >:: = SIN <expresie>

<funcţia COS>:: = COS <expresie>

<funcţia TAN>:: = TAN <expresie>

unde <expresie> este o expresie prin a cărei evaluare rezultă o constantă de tip numeric.

ASN, ACS, ATN

Funcţiile ASN, ACS, ATN permit calcularea unor aproximaţii respectiv pentru valorile.**arcsin x, arccos x, arctg x** unde **x** este valoarea argumentului indicat.

Sintaxa utilizării acestor funcţii este:

<funcţia ASN>:: = ASN <expresie>

<funcţia ACS) :: = ACS <expresie>

<funcţia ATN>:: = ATN (<expresie>

unde <expresie> este o expresie prin a cărei evaluare rezultă o constantă de tip numeric.

Exemple. ASN 0.5 este 0.52359878
ACS 0.5 este 1.0471976
ACS −1 este 3.1415927
ATN 1 este 0.78539816
ATN 0 este 0

Apelarea funcţiilor ASN, ACS pentru valori ale argumentului ce nu aparţin intervalului [−1,1] conduce la afişarea mesajului de eroare "A Invalid argument" .

Exemplu. Apelările ASN 1.5, ACS −1.2 conduc la mesajul de eroare "A Invalid argument".

3.4 Funcţii speciale: EXP, LN, PI, RND, PEEK

EXP

Funcţia EXP necesită un singur argument ce poate fi orice expresie prin a cărei evaluare rezultă o constantă de tip numeric.

<funcţia EXP> :: = EXP <expresie>

Dacă **x** este valoarea obţinută prin evaluarea expresiei indicată ca argument, atunci rezultatul apelării funcţiei EXP <expresie> este o aproximaţie a numărului e^x.

Exemple. EXP 0 este 1
EXP 1 este 2.7182818
EXP −1 este 0.36787944

LN

Funcţia LN permite calcularea unei aproximaţii a numărului **ln x.** Sintaxa utilizării funcţiei LN este

<funcţia LN> :: = LN <expresie>

unde <expresie> este o expresie prin a cărei evaluare rezultă o constantă numerică pozitivă.

Exemple. LN 2 este 0.69314718
LN EXP 0 este 0
LN EXP 1 este 1

Apelarea funcţiei LN pentru argumente ce nu sînt numere pozitive conduce la afişarea unui mesaj de eroare.

Exemplu. Apelările LN 0, LN −1 conduc la mesajul de eroare "A Invalid argument".

PI

Funcţia PI nu necesită nici un argument .

<funcţia PI>:: = PI

Rezultatul apelării funcţiei PI este obţinerea unei aproximaţii pentru π şi anume a valorii 3.14159265.

Exemple. PI este 3.14159265
LN PI este 1.1447299
PI ↑ 2 este 9.8696044
SQR PI este 1.7724539

Observaţie. Reprezentarea numerelor în calculator ca şi efectuarea operaţiilor aritmetice este în general "inexactă" în sensul că prin reţinerea numai a unui număr finit de zecimale se obţin în general numai aproximaţii pentru valorile teoretice corespunzătoare.

Astfel se explică situaţii de tipul:

valoarea expresiei 7 ↑ 3 = 7*7*7 este 0 (deci egalitatea nu este adevărată).

Mai mult decît atît, deoarece afişajul constantelor se realizează cu 9 (sau 10 cifre), deşi valorile calculate prin apelările EXP 3 şi EXP 1 ↑ 3 sînt ambele egale cu 20.085537, valoarea expresiei EXP 3 = EXP 1 ↑ 3 este 0 (egalitatea nu este adevărată).

RND

Funcţia RND nu necesită argumente.

<funcţia RND>:: = RND

rezultatul apelării fiind obţinerea unui număr pseudoaleator din intervalul (0,1).

Ca în cazul tuturor tehnicilor de generare de numere aleatoare, pe calculator, este improprie folosirea termenului de "aleator", numerele generate fiind mai curînd numere "aproape" aleatoare (pseudoaleatoare) decît "aleatoare" în sensul strict matematic.

În cazul microcalculatoarelor, numerele generate prin apelarea funcţiei RND aparţin unei secvenţe fixate cuprinzînd 65536 de componente. Numerele din secvenţă sînt obţinute prin calcularea succesivă a puterilor lui 75, pentru fiecare număr astfel calculat se reţine restul obţinut prin împărţirea lui la 65537, din care se scade 1 şi, în continuare, se împarte numărul rezultat la 65536. Apelări succesive ale funcţiei RND determină obţinerea pe rînd a numerelor din secvenţă. La fiecare resetare a microcalculatorului sau după executarea unei comenzi NEW se revine la primul număr din secvenţă.

PEEK

Funcţia PEEK permite obţinerea informaţiei conţinute într-un anumit byte al memoriei.

<funcţia PEEK>:: = PEEK (<expresie>)

unde <expresie> este o e xpresie prin a cărei evaluare se obţine o valoare numerică, fie aceasta x. Dacă n este valoarea obţinută prin aplicarea operaţiei de rotunjire a numărului x, atunci trebuie ca 0≤ n ≤ 65535

Dacă <expresie> este o expresie reprezentată printr-o singură variabilă sau o singură constantă, atunci parantezele pot fi eventual omise.

Exemplu.

```
PEEK 256
PEEK x
PEEK (256*x)
```

4. Instrucţiuni şi comenzi BASIC

4.1 Atribuire: LET

LET (Instrucţiunea de atribuire)

Sintaxa instrucţiunii de atribuire în BASIC este

<instrucţiunea de atribuire>:: =

{<etichetă>} LET <variabilă> = <expresie>

Instrucţiunea de atribuire poate fi utilizată şi drept comandă.

Executarea unei instrucţiuni de atribuire constă în evaluarea expresiei

<expresie> şi atribuirea ca valoare curentă pentru variabila <variabilă> a constantei rezultate în urma evaluării.

În utilizarea instrucţiunii (comenzii) LET trebuie avută în vedere condiţia ca rezultatul obţinut prin evaluarea expresiei date să fie de acelaşi tip cu tipul corespunzător variabilei (ambele de tip numeric respectiv ambele de tip string).

Exemple.

```
LET a = 1
10 LET a = 2:LET b = 3: LET a = a*b
LET a$ = "abcd"
```

```
10 LET a$ = "a+2": LET a = 3: LET b =VAL a$
```

Observaţie. Există variante de BASIC în care instrucţiunea de atribuire este unica instrucţiune care nu este caracterizată de un cuvînt cheie. In cazul unei astfel de variante instrucţiunea

LET a=3 se scrie a = 3

4.2 Afişaj: PRINT, AT, TAB, OVER

PRINT

Sintaxa instrucţiunii PRINT este

<instrucţiunea PRINT> :: = {<etichetă>}PRINT<listă>{[;,]}

unde <listă> reprezintă o secvenţă de expresii, eventual opţiuni, componentele secvenţei fiind separate printr-unul din separatorii ";" sau "," .

Rezultatul executării unei instrucţiuni PRINT este afişarea pe ecran, conform opţiunilor care sînt formulate, a rezultatului evaluării expresiilor care figurează în <listă>.

Observaţie. Accesarea ecranului se poate realiza în două maniere de lucru: în modul de lucru cu putere rezolutivă mică şi în modul de lucru cu putere mare rezolutivă.

În cadrul acestui paragraf vom considera în exclusivitate **modul de lucru în rezoluţie de mică putere.**

În acest mod de lucru, ecranul este partajat în 24x32 celule, dispuse pe 24 linii şi 32 coloane, în fiecare celulă putînd fi înscris un singur caracter. Referirea la o anumită celulă a ecranului se realizează prin indicarea numerelor indicînd linia şi respectiv coloana corespunzătoare celulei vizate.

Numerotarea liniilor şi a coloanelor se realizează prin numerele naturale 0..23 pentru linii, respectiv 0..31 pentru coloane, originea fiind considerată colţul din stînga, sus al ecranului (celula corespunzătoare colţului din stînga sus al ecranului are "coordonatele" egale cu 0). Pentru afişarea informaţiei, sînt disponibile numai liniile de la 0 la 21, spaţiul de pe ecran corespunzător liniilor 22 şi 23 fiind rezervat pentru afişarea de mesaje, comunicare între utilizator şi calculator etc.

Semnificaţia utilizării separatorilor ";" respectiv "," .

Dacă două componente ale listei PRINT sînt separate prin ';', atunci înscrierea informaţiei celei de a doua componente se realizează începînd din celula de pe aceeaşi linie imediat la dreapta ultimei celule utilizate pentru afişarea informaţiei primei componente (dacă scrierea informaţiei primei componente s-a terminat în ultima celulă a unei linii, atunci afişarea celei de a doua componente va avea loc începînd din coloana 0 a liniei următoare). Utilizarea separatorului "," între două

componente ale unei liste PRINT determină înscrierea a 12 caractere blank (spații libere) între ultima celulă utilizată în afișarea informației corespunzătoare primei componente și prima celulă din care începe afișarea informației corespunzătoare celei de a doua componente (cu trecerea eventual pe linia următoare). O instrucțiune PRINT poate fi opțional terminată printr-unul din acești separatori. In cazul în care nu este prezent nici unul dintre acești separatori la sfîrșitul unei instrucțiuni PRINT, executarea următoarei instrucțiuni PRINT (dacă există) va determina afișarea informației începînd cu linia imediat următoare(conform cu eventualele opțiuni prezente în cadrul instrucțiunii PRINT respective). Dacă o instrucțiune PRINT se încheie cu unul din separatorii ";" sau "," atunci executarea următoarei instrucțiuni PRINT va determina afișarea începînd din celula imediat următoare ultimei celule utilizate pentru afișaj (dacă este utilizat ";")respectiv inserînd 12 spații libere (dacă este utilizat ","), bineînțeles cu posibilitatea trecerii la linia următoare.

Dacă parametrul <lista> lipsește într-o instrucțiune PRINT, atunci prin executarea unei astfel de instrucțiuni se va realiza trecerea la linia următoare, respectiv scrierea unei linii de blank-uri (în funcție de instrucțiunea PRINT precedentă care s-a încheiat sau nu cu unul dintre separatorii ";" respectiv ",").

Exemple.

a) Prin executarea programului

```
10  LET a=2: LET b=a↑3 : PRINT "a="; a ;"¤¤¤";
"b=¤"; b ; "¤¤¤"; "a+b="; a+b
```

va rezulta afișarea pe ecran a informației

```
a=2¤¤¤b=8¤¤¤a+b=10
```

b) Același rezultat se poate obține și prin executarea următoarei variante de program

```
10 LET a$ = "¤¤¤" : LET a=2: LET b = a↑3:
   PRINT "a=" ;a; a$; "b="; b; a$; "a+b="; a+b
```

c) Prin executarea programului

```
10 LET a=3 : LET b=5
20 PRINT 1,2: PRINT a=b; a<b; a+b
```

va rezulta afișarea pe ecran a informației

```
          1              2
         01              8
```

deoarece valorile curente pentru variabilele a,b sînt 3 respectiv 5, evaluarea expresiilor a=b, a<b va conduce la valorile 0 respectiv 1 (au fost utilizați separatorii ';' și ',').

d) După executarea programului

```
10 PRINT 1,2; : PRINT 3; 4;
20 PRINT 5;6;7
```

pe ecran va apare

```
 1                 234567
```

Observaţie. O particularitate a limbajului BASIC care se dovedeşte a fi extrem de utilă în depanarea programelor este aceea că după executarea unui program accesul la valorile variabilelor (atît pentru variabilele simple, cît şi pentru variabilele indexate) este permis şi prin comenzi (instrucţiuni care nu fac parte din program).

Să considerăm următorul exemplu. Presupunem că a fost introdus în memoria calculatorului (dar nu a fost executat încă!) următorul program

```
10 LET a =3
```

Formulînd comanda PRINT a, se va obţine mesajul de eroare "Variable not found". Dacă se solicită executarea programului şi apoi se formulează comanda PRINT a, atunci pe ecran va fi afişat 3, care reprezintă valoarea curentă a variabilei (din program) avînd identificatorul **a**.

Pentru controlul afişării informaţiei pe ecran prin utilizarea instrucţiunii PRINT este posibilă formularea unor opţiuni permiţînd selectarea celulei din care trebuie realizată scrierea informaţiei corespunzătoare unei componente de tip expresie din listă, cromatica utilizată (INK, PAPER, BORDER utilizate ca opţiuni), efectele sonore (BEEP utilizată ca opţiune) precum şi calităţi solicitate pentru gradul de luminozitate cu care apare înscrisă o anumită informaţie pe ecran (FLASH, BRIGHT utilizate ca instrucţiuni).

Vom prezenta aici numai opţiunile ce permit accesarea unei anumite celule a ecranului (opţiunile AT şi TAB).

AT

Opţiunea AT poate figura drept componenta a parametrului <lista> corespunzător unei instrucţiuni PRINT conform cu sintaxa

<opţiune AT>:: = AT <expresie 1>, <expresie 2>

unde <expresie 1>, <expresie 2> sînt expresii prin ale căror evaluări se obţin valori de tip numeric. Fie **x,y** valorile obţinute prin evaluarea respectiv a celor două expresii şi **n,p** valorile rezultate prin aplicarea operaţiei de rotunjire a numerelor **x** şi respectiv **y**.

In formularea opţiunii **AT** trebuie respectate condiţiile $0 \leq n \leq 21$, $0 \leq p \leq 31$.

Prezenţa opţiunii **AT** drept componentă a listei unei instrucţiuni **PRINT** determină controlul afişajului pe ecran să fie poziţionat în dreptul celulei de pe linia **n** şi coloana **p**.

Dacă opţiunea **AT** în lista este separată de următoarea componentă prin ";" atunci afişarea informaţiei corespunzătoare acestuia va începe din celula de coordonate **n,p**. Deşi posibilă este nejustificată în general utilizarea separatorului "," după opţiunea **AT**. Efectul în acest caz este de a deplasa dispozitivul de scriere pe

ecran în dreptul celulei de coordonate **n,p**, se inseră 12 spaţii (cu trecerea eventual la linia următoare) din celula la care s-a ajuns începînd înscrierea informaţiei corespunzătoare următoarei componente din listă.

Formularea unor opţiuni AT consecutive într-o listă PRINT este posibilă, dar nejustificată.

Exemple. a) Prin executarea programului

```
10 LET a$ = "abcdefgh"
20 PRINT AT 3,1;a$ (3 TO);AT 3,20;
   a$ (1 TO 3);AT 3,12;a$ (3)
```

va rezulta pe cea de a 4-a linie a ecranului (numărul corespunzător liniei fiind 3) informaţia afişată.

¤cdefgh¤¤¤¤¤ c¤¤¤¤¤¤¤abc

unde prin '¤'a fost i..dicată o celulă conţinînd un blank.

b) Prin executarea următoarelor variante de program

```
 I) 10 PRINT AT 3.5,1.2; "A"; AT 3.2,1.5; "B";
        AT 3.6,1.6; "C"
II) 10 PRINT AT 4,1; "A"; AT 3,2; "B" ;
        AT 4,2; "C"
```

se obţin pe ecran aceleaşi rezultate

	coloana 0	coloana 1	coloana 2
linia 0			
linia 1			
linia 2			
linia 3			B
linia 4		A	C

TAB

Opţiunea TAB poate figura drept componenta a unei liste PRINT conform cu sintaxa

<opţiunea TAB>:: = TAB <expresie>

unde <expresie> este o expresie prin a cărei evaluare rezultă o constantă numerică. Fie **n** rezultatul obţinut prin aplicarea operaţiei de rotunjire a valorii rezultate prin evaluarea expresiei indicate. Formularea opţiunii TAB trebuie astfel făcută încît să rezulte **n**≥0. Fie p restul împărţirii lui **n** la 32.

Efectul prezenţei opţiunii TAB este de a determina ca dispozitivul de control al scrierii informaţiei pe ecran să se poziţioneze în dreptul coloanei **p** de pe aceeaşi linie sau de pe linia următoare şi anume:

- dacă pînă la întîlnirea opţiunii TAB din lista PRINT informaţia înscrisă pe ecran nu depăşeşte coloana **p**, atunci TAB determină poziţionarea controlului de afişaj

în dreptul coloanei **p** pe aceeaşi linie înscriind blank-uri în toate poziţiile intermediare urmînd ca afişarea în continuare să fie realizată din celula următoare (eventual se trece pe linia următoare dacă după opţiunea TAB este utilizat separatorul ";" respectiv înserarea a 12 spaţii libere dacă este utilizat separatorul ",").

- dacă pînă la întîlnirea opţiunii TAB din lista PRINT informaţia afişată pe linia curentă depăşeşte coloana **p**, atunci se trece la linia următoare în primele **p** coloane sînt inserate blank-uri urmînd ca în continuare informaţia să fie afişată începînd din coloana **p+1** sau inserînd încă 12 spaţii libere după cum după opţiunea TAB apare separatorul ";" sau ",".

Este important de reţinut faptul că poziţionarea la o anumită celulă a ecranului prin utilizarea opţiunii AT nu presupune inserarea de blank-uri pe poziţiile intermediare, în timp ce selectarea coloanei prin opţiunea TAB este realizată prin inserarea de blank-uri în celulele intermediare.

Exemple. a) Prin executarea programului

```
10 PRINT "abcdefghijklm"; TAB 3; "123"
```

se obţine pe ecran

```
abcdefghijklm
   123
```

b) Prin executarea programului

```
10 PRINT "abcdefghijklm"; AT 0,5; TAB 10
```

se obţine pe ecran

```
abcde¤¤¤¤¤klmn
```

Explicaţia rezultă din faptul că, pe de o parte, înscrierea unui simbol într-o celulă în care este deja înscris un simbol implică ştergerea vechiului conţinut şi înscrierea noului conţinut, pe de altă parte, selectarea unei anumite celule cu opţiunea AT nu afectează informaţia din celulele intermediare, respectiv selectarea unei anumite coloane cu opţiunea TAB implică inserarea de blank-uri în toate celulele intermediare.

```
10 PRINT AT 3,1;"abc"; AT 3,19;
   "123456"; AT 3,2; TAB 17; "ABCD"
```

va rezulta afişarea pe ecran

```
¤a¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤ABCD 3456
```

OVER

Opţiunea OVER poate figura ca opţiune în cadrul unei liste PRINT într-una din modalităţile următoare

```
OVER 0 ;
OVER 1 ;
```

Aşa după cum s-a precizat anterior, tentativa de înscriere a unui caracter într-o altă celulă a ecranului în care este deja înscris un caracter determină ştergerea vechiului caracter şi înscrierea celui nou.

Prezenţa opţiunii OVER 1; în cadrul unei liste PRINT determină ca în cadrul afişajului realizat prin instrucţiunea respectivă corespunzător expresiilor care urmează în listă acestei opţiuni, tentativa de înscriere a unui caracter într-o celulă în care este deja înscris un caracter să conducă la suprapunerea celor două caractere. Opţiunea OVER 0 anulează efectul opţiunii OVER 1. Este de reţinut faptul că prezenţa unei opţiuni OVER într-o listă PRINT are efect numai asupra afişajului determinat de această instrucţiune. Următoarele exemple ilustrează efectul opţiunii OVER 1 şi pune în evidenţă mai bine efectele opţiunilor AT şi TAB.

Exemple.

```
10 PRINT AT 3,1; "abcdefgh": AT 3,5; "1"
```

se afişează pe linia 3

```
abcd1fgh
```

```
10 PRINT AT 3,1; "abcdefgh"; OVER 1; AT 3,5; "1"
```

se afişează

```
abcdefgh
```

unde caracterele e şi 1 apar suprapuse în celula cu linia 3 şi coloana 5.

```
10 PRINT "abcdefghijklmn"; AT 0,5; TAB 10
```

se afişează

```
abcde¤¤¤¤¤klmn
```

```
10 PRINT "abcdefghijklmn" ; AT 0,5; OVER 1; TAB 10
```

se afişează

```
abcdefghijklmn
```

4.3 Citire: INPUT

INPUT

Un program BASIC reprezintă codificarea unei secvenţe de operaţii şi acţiuni prin care se realizează descrierea etapelor unui algoritm pentru rezolvarea unei probleme date. Datele reprezentînd informaţia prelucrată de algoritm pot fi furnizate calculatorului, fie în timpul execuţiei programului, fie introduse în program drept componente ale unui bloc de date şi preluate pe măsură ce este necesară prelucrarea lor.

Instrucţiunea INPUT este utilizată pentru introducerea datelor de prelucrat prin tastatură în faza de execuţie a unui program.

Sintaxa instrucţiunii INPUT este

<instrucţiunea INPUT> :: = { <etichetă> } INPUT <listă>

unde <listă> este o secvenţă ale cărei componente sînt separate prin "," sau ";". Componentele unei liste INPUT pot fi variabile simple, variabile indexate atît de tip numeric cît şi de tip string, constante de tip string, precum şi structuri de tipul (<sublistă>).

Executarea unei instrucţiuni INPUT determină calculatorul să aştepte ca utilizatorul să introducă valori pentru variabilele din lista INPUT. Introducerea fiecărei valori se realizează prin tastarea valorii respective urmată de acţionarea tastei ENTER.

Valorile astfel introduse sînt atribuite ca valori curente variabilelor din lista INPUT.

Exemplu. `INPUT a,b$,c` determină calculatorul să aştepte ca utilizatorul să indice prin tastatură cîte o valoare pentru fiecare din cele trei variabile din listă. Presupunînd că utilizatorul a indicat valorile 1 ENTER a+b ENTER 3 ENTER

execuţia instrucţiunii INPUT este încheiată efectul fiind acela că pentru **a** este asociată valoarea 1, pentru b$ constanta string "a+b", iar pentru variabila **c** valoarea 3.

Prezenţa componentelor de tipul constantă string într-o listă INPUT asigură posibilitatea unui gen conversaţional de programare deoarece constantele string sînt afişate pe ecran.

Exemplu.

```
10 INPUT "dati valoarea pentru a"; c ,
   "dati valoarea pentru b"; b
```

Executarea acestui program va determina afişarea textului `"dati valoarea pentru a"` după care calculatorul aşteaptă introducerea unei valori pe care o atribuie ca valoare variabilei **c**. În continuare afişează textul `"dati valoarea pentru b"` după care aşteaptă ca utilizatorul să introducă prin tastatură un număr pe care-l atribuie ca valoare variabilei **b**. Astfel se crează posibilitatea informării utilizatorului asupra datelor pe care trebuie să le introducă, facilitate deosebit de utilă ţinînd cont de faptul că într-un program pot exista oricîte instrucţiuni INPUT.

Observaţie. Pentru o anumită categorie de minicalculatoare nu este posibilă utilizarea constantelor string drept componente ale unei liste INPUT decît o singură dată în fiecare listă şi anume înainte de prima componentă de tip variabilă. Într-o astfel de situaţie se poate asigura, totuşi, un stil de programare conversaţional prin utilizarea eventual a mai multor instrucţiuni INPUT.

Observaţie. O particularitate interesantă o reprezintă faptul că pentru variabilele dintr-o listă INPUT se pot da ca valori expresii în care pot să apară nume de variabile definite (care au asociate respectiv valori). Calculatorul evaluează ex-

presiile date şi valorile rezultate le atribuie ca valori respectiv variabilelor din lista INPUT.

Exemplu.

```
10 LET a = 3: LET b=2
20 INPUT "dati valoarea pentru b", b
30 PRINT b
```

Dacă în momentul executării instrucţiunii INPUT, deci după ce a fost afişat mesajul 'daţi valoarea pentru **b**' se tastează expresia **a + b** atunci rezultatul afişat ca efect al executării instrucţiunii PRINT din program va fi 5.

Structurile de tipul (<sublistă>) prezente într-o listă INPUT determină afişarea informaţiei corespunzătoare parametrului <sublistă>. Parametrul <sublistă> poate fi orice secvenţă de expresii componentele fiind separate prin ";" sau ",".

Expresiile componente ale unei astfel de subliste sînt evaluate şi valorile rezultate sînt afişate pe ecran controlul asupra modului de afişaj fiind realizat prin utilizarea separatorilor ";" şi "," cu aceleaşi semnificaţii ca şi în cazul listelor PRINT.

Exemple.

```
a) 10 LET a=3: LET b= a↑2
    20 INPUT ("valoarea curenta a lui b este =" ; a↑2),
       "dati noua valoare pentru b", b
```

se afişează

```
valoarea curenta a lui b este = 9
dati noua valoare pentru b
```

după care se aşteaptă noua valoare ce va fi atribuită variabilei b.

```
b) 10 INPUT "ce virsta aveti ?", v
    20 INPUT ("virsta dumneavoastra este "; v;"."),
       "ce virsta are colegul dumneavoastra ?"; t
```

Programul considerat este echivalent cu

```
10 INPUT "ce virsta aveti?", v
20 PRINT "virsta dumneavoastra este "; v;".",
30 INPUT"ce virsta are colegul dumneavoastra?"; t
```

Utilizarea acestor facilităţi permite scrierea comodă de programe în care calculatorul "poate conversa" cu utilizatorul, ceea ce asigură nu numai o programare uşoară şi "plăcută", dar îşi dovedeşte eficienţa într-o serie de aplicaţii.

4.4 Control: GO TO, IF-THEN, FOR-NEXT

GO TO

Instrucţiunea GO TO permite întreruperea execuţiei secvenţiale a instrucţiunilor dintr-un program. Efectul unei instrucţiuni GO TO constă în determinarea unui "salt" la o linie de program selectată.

Sintaxa instrucţiunii GO TO este

<instrucţiunea GO TO> :: = {<etichetă>} GO TO <expresie>

unde <expresie> este o expresie prin a cărei evaluare rezultă o constantă de tip numeric.

Executarea instrucţiunii GO TO se realizează prin evaluarea expresiei considerate şi determinarea "saltului" la linia de program avînd numărul de ordine **n**, unde **n** este cel mai mic număr de linie program cu **n** mai mare sau egal decît valoarea obţinută pentru <expresie>, în continuare fiind executate instrucţiunile componente ale acestei linii de program.

IF-THEN

Instrucţiunea IF-THEN permite selectarea unei decizii în funcţie de valoarea corespunzătoare unei anumite expresii.

Anumite variante de BASIC dispun de două tipuri de instrucţiuni IF, şi anume, IF-THEN şi respectiv IF-THEN-ELSE.

Sintaxa corespunzătoare instrucţiunilor IF-THEN (IF-THEN-ELSE) poate fi descrisă astfel:

<instrucţiunea IF-THEN > :: =

{<etichetă>}IF<expresie> THEN<instrucţiune>

<instrucţiunea IF-THEN-ELSE> :: =

{ <etichetă>} IF <expresie>

THEN <instrucţiune 1> ELSE <instrucţiune 2>

unde <expresie> reprezintă o expresie prin a cărei evaluare rezultă o constantă de tip numeric, iar <instrucţiune>, <instrucţiune 1>, <instrucţiune 2> sînt instrucţiuni BASIC.

Exemple.

```
10 IF a=b THEN LET a = a*b
IF a<b THEN IF a>= 2 THEN LET b = a*b
IF a THEN PRINT a
IF a=b THEN LET a=0 ELSE LET b=3
```

Executarea unei instrucţiuni de tipul IF-THEN-ELSE constă în evaluarea expresiei indicate şi dacă valoarea obţinută este ≠0, se execută instrucţiunea <instrucţiune 1>,

altfel se execută în continuare instrucţiunea <instrucţiune 2>.

Deoarece, în general, pentru variantele de BASIC pentru microcalculatoare nu este disponibilă decît instrucţiunea de tip IF–THEN, în continuare ne vom referi în exclusivitate la aceasta.

Executarea instrucţiunii IF–THEN constă în evaluarea expresiei indicate şi dacă valoarea rezultată este diferită de zero, atunci se execută instrucţiunea indicată după cuvîntul cheie THEN şi în continuare toate instrucţiunile ce urmează instrucţiunii IF–THEN aflate pe linia respectivă de program. Dacă valoarea rezultată prin evaluarea expresiei este egală cu zero, atunci este abandonată întreaga linie de program şi se continuă cu executarea primei instrucţiuni din linia următoare de program.

Exemplu.

```
10 LET a=3
20 IF a THEN PRINT "A"
30 LET a=0: IF a THEN PRINT "B"
40 LET a=-1: IF a THEN PRINT "C"
50 LET a=0.2: IFa THEN PRINT "D"
```

se va afişa pe ecran

```
A
C
D
```

Instrucţiunea IF–THEN poate fi utilizată atît ca instrucţiune program cît şi sub forma unei comenzi (independente de program) adresate calculatorului.

Observaţie. Anumite variante de BASIC (de exemplu, BASIC–PLUS–2 utilizat frecvent pe minicalculatoare de tip CORAL) realizează executarea instrucţiunii IF–THEN în modul următor: dacă valoarea obţinută prin evaluarea expresiei este diferită de zero, atunci se execută instrucţiunea ce urmează cuvîntului cheie THEN şi în continuare toate instrucţiunile ce urmează instrucţiunii IF–THEN de pe linia de program respectivă. Dacă valoarea rezultată este 0, atunci nu se execută instrucţiunea <instrucţiune> şi se continuă cu executarea primei instrucţiuni ce urmează instrucţiunii IF–THEN pe aceeaşi linie de program.

Această particularitate a variantei de BASIC cu care se lucrează este foarte important să fie cunoscută de către utilizator, ignorarea ei putînd să conducă la erori în scrierea programelor.

Pentru facilitarea înţelegerii diferenţei dintre aceste două moduri complet diferite în care poate fi executată instrucţiunea IF–THEN convenim să delimităm domeniul de acţiune al instrucţiunii IF marcînd sfîrşitul domeniului cu FI.

În cazul primului tip de execuţie delimitatorul FI corespunde sfîrşitului liniei de program din care face parte instrucţiunea IF–THEN considerată, respectiv în cazul celui de al doilea tip de execuţie delimitatorul FI corespunde sfîrşitului instrucţiunii imediat următoare cuvîntului cheie THEN. In cazul obţinerii unei valori nenule pentru expresia indicată se execută toate instrucţiunile care urmează cuvîntului cheie THEN, în caz contrar se va executa prima instrucţiune aflată după

delimitatorul FI. Desigur, prezența unei instrucțiuni GO TO sau GO SUB ca primă instrucțiune după cuvîntul cheie THEN determină ca în cazul obținerii prin evaluarea expresiei a unei valori diferite de zero abandonarea liniei de program curente (ca efect al executării instrucțiunii GO TO respectiv GO SUB).

Instrucțiunea ce urmează cuvîntului cheie THEN poate fi eventual tot o instrucțiune de tip IF-THEN.

Exemple.

```
A)10 LET a=1 : LET b=1
  20 IF a=b THEN IF a=1 THEN PRINT a: PRINT "DA"
  30 PRINT "SFIRSIT"
```

se obțin rezultatele

```
   1
   DA
   SFIRSIT
```

```
B)10 LET a=1 : LET b=1
  20 IF a=b THEN IF a=2 THEN PRINT a: PRINT "DA"
  30 PRINT "SFIRSIT"
```

conduce la rezultatul

```
   SFIRSIT
```

```
C) Prin executarea programului
   10 LET a=1 : LET b=1
   20 IF a=b THEN IF a=1 THEN IF b=1 THEN PRINT a:
      PRINT b: PRINT "DA"
   30 PRINT "SFIRSIT"
```

se obține pe ecran

```
  1
  1
  DA
  SFIRSIT
```

D) Următorul exemplu sugerează o posibilitate "conversațională" de reluare a execuției unui program dacă utilizatorul dorește aceasta.

```
 5 LET b$ = " "
10 INPUT "dati valorile pentru a,b,c",a,b,c
20 PRINT "expresia a+b*c are valoarea a="; a+b*c
30 INPUT ("valorile pentru a,b,c sint respectiv a=";
   a; b$; "b="; b; b$; "c=";c," doriti reluarea ?
   (raspundeti da sau nu)"); a$
40 IF a$ = "da" THEN GO TO 10
50 PRINT "SFIRSIT"
```

• P1

Program pentru discuţia şi rezolvarea unei ecuaţii de gradul 2:
$ax^2 + bx + c = 0$.

```
 10 INPUT "Dati valorile coeficientilor a="; b, "c=";c
 20 IF a=0 THEN GO TO 90
 30 LET delta = b*b - 4*a*c
 40 IF delta <0 THEN GO TO 70
 50 LET x1 = (-b+SQR delta)/(2*a):
    LET x2 = (-b-SQR delta)/(2*a)
 60 PRINT "Ecuatia are radacini reale":
    PRINT:PRINT "x1=";x1,"x2="; x2:
    CLS : GO TO 130
 70 LET r = -b/(2*a): LET i = SQR (-delta)/(2*a)
 80 PRINT "Ecuatia are radacini complexe":PRINT:
    PRINT "x1="; r; "+i"; i,"x2=" ; r; "-i"; i:
    CLS : GO TO 130
 90 IF b<>0 THEN PRINT "Ecuatia este de gradul 1":
    PRINT: PRINT "X="; -c/b: CLS: GO TO 130
100 PRINT "Ecuatia este": PRINT:PRINT c; "=0","deci",
110 IF c≠0 THEN PRINT "probabil ca ati gresit": CLS:
    GO TO 130
120 PRINT "orice x real este solutie"
130 INPUT "Doriti reluarea programului ?(da/nu)",a$
140 IF a$ = "da" THEN GO TO 10
150 PRINT "LA REVEDERE!"
```

FOR-NEXT

Instrucţiunea FOR-NEXT permite reprezentarea în limbajul BASIC a structurilor repetitive ce presupun executarea unui grup de instrucţiuni pentru diferite valori aflate în progresie aritmetică, ale unei variabile numită **variabilă de control**.

Identificatorul utilizat pentru o variabilă utilizată ca variabilă de control asociată unei "bucle" FOR-NEXT trebuie să fie reprezentat printr-o singură literă. Termenul de "buclă" FOR-NEXT este justificat de modul în care se execută această instrucţiune.

Instrucţiunea FOR-NEXT se compune de fapt din două instrucţiuni respectiv instrucţiunea FOR şi instrucţiunea NEXT care eventual pot fi situate pe linii diferite de program. Este obligatoriu însă ca execuţia instrucţiunii FOR să preceadă execuţia instrucţiunii NEXT asociate. In general, logica programului solicită prezenţa instrucţiunii NEXT pe o linie de program de număr de ordine mai mare decît sau egal cu cel corespunzător liniei de program pe care se află instrucţiunea FOR asociate. Sintaxa instrucţiunii FOR este

<instrucţiunea FOR> ::=

{<etichetă>}FOR<variabilă> = <expresie 1>TO <expresie 2>

{STEP <expresie 3>}

unde <variabilă> este identificatorul utilizat ca nume pentru variabila de control (constînd dintr-o singură literă), <expresie 1>, <expresie 2>, <expresie 3> fiind expresii prin ale căror evaluări rezultă constante de tip numeric.

Sintaxa instrucţiunii NEXT este

<instrucţiunea NEXT>:: = {<etichetă>} NEXT <variabilă>

unde <variabilă> este identificatorul utilizat ca nume pentru variabila de control (acelaşi identificator ca în cazul parametrului <variabilă> în instrucţiunea FOR asociată).

Exemple.

```
A)  10 LET x=0 : FOR a=1 TO 5 STEP 3.2
    20 LET x=x+a: NEXT a: PRINT x
B)  10 LET x=0: FOR a=1 TO-1 STEP-1/2:
       LET x=x+a: PRINT x: NEXT a
```

Executarea unei instrucţiuni FOR-NEXT constă în iniţierea unor acţiuni începînd cu evaluarea expresiilor <expresie 1>, <expresie 2>, <expresie 3>.

Dacă parametrul (opţional după cum rezultă din descrierea sintaxei instrucţiunii FOR) STEP <expresie 3> lipseşte, atunci calculatorul presupune ca valoare pentru <expresie 3> constanta 1.

Se iniţiază un control asupra "corectitudinii logice" a instrucţiunii FOR–NEXT în sensul următor :

Pentru simplificare, să notăm cu **v** identificatorul utilizat ca nume pentru variabila de control şi cu **init, fin, pas** respectiv valorile obţinute prin evaluarea expresiilor <expresie 1>, <expresie 2>, <expresie 3>.

Dacă **pas**>0 şi **init**<**fin**, atunci se atribuie **init** ca valoare pentru **v** şi se execută instrucţiunile începînd cu prima instrucţiune după FOR. Executarea instrucţiunii NEXT asociate determină ca valoarea variabilei de control să fie modificată, noua valoare fiind obţinută din valoarea curentă la care se însumează valoarea **pas.** Dacă noua valoare este mai mică sau egală cu **fin,** atunci se reia executarea instrucţiunilor începînd cu instrucţiunea imediat următoare instrucţiunii FOR. Dacă noua valoare pentru **v** este mai mare decît **fin**, atunci se execută instrucţiunea imediat următoare instrucţiunii NEXT.

Dacă **pas**>0 şi **init**>**fin**, atunci se execută în continuare prima instrucţiune situată după instrucţiunea NEXT (de pe aceeaşi linie de program).

Dacă **pas**<0 şi **init**<**fin**, atunci se execută prima instrucţiune situată după instrucţiunea NEXT. Dacă **pas**<0 şi **init**>fin atunci se atribuie lui **v** valoarea **init**, se execută instrucţiunile începînd cu prima instrucţiune după FOR, executarea instrucţiunii NEXT asociate constînd în modificarea valorii variabilei de control prin însumarea cu valoarea curentă a valorii **pas** şi verificarea dacă noua valoare este mai mare sau egală cu **fin**. In caz afirmativ se iniţiază reluarea executării instrucţiunilor începînd cu instrucţiunea imediat următoare instrucţiunii FOR, altfel se execută instrucţiunea imediat următoare (pe aceeaşi linie de program)

instrucţiunii NEXT.

Dacă **init** = **fin,** atunci indiferent dacă **pas**>0 sau **pas**<0 se realizează executarea o singură dată a instrucţiunilor ce compun corpul buclei FOR–NEXT.

Dacă **init** = **fin** şi **pas** = 0 atunci se realizează o ciclare infinită (nu se părăseşte bucla FOR–NEXT niciodată). Dacă **init>fin** şi **pas**=0, atunci se execută instrucţiunea imediat următoare instrucţiunii NEXT.

Bineînţeles, prezenţa unei instrucţiuni GO TO printre instrucţiunile ce compune corpul unei bucle FOR–NEXT determină părăsirea buclei înainte să fi fost îndeplinită condiţia de părăsire "normală" a buclei (cu condiţia ca instrucţiunile din bucla FOR–NEXT să fie executate cel puţin o dată).

Trebuie reţinut că, după părăsirea buclei FOR–NEXT (prin executarea unei instrucţiuni GO TO sau cînd este îndeplinită condiţia de ieşire) valoarea curentă a variabilei de control poate fi utilizată în continuare în program.

Exemple.

a)
```
10 FOR i=1 TO 10
30 NEXT i
40 PRINT i
```

va determina afişarea valorii 11.

Executarea programului

b)
```
10 FOR i=1 TO 10
20 IF i>5 THEN GO TO 40
30 NEXT i
40 LET a=i: PRINT a
```

va determina afişarea valorii 6.

c)
```
10 FOR i=1 TO 10 STEP-1
20 PRINT i
30 NEXT i: PRINT "SFIRSIT"
```

are ca rezultat

```
SFIRSIT
```

d)
```
10 FOR i=1 TO 10 STEP 0
20 PRINT i
30 NEXT i
```

determină afişarea indefinit a valoarii 1 (execuţia programului se suspendă eventual prin acţionarea tastei BREAK).

e)
```
10 FOR i=1 TO - 10 STEP 0
20 PRINT i
30 NEXT i : PRINT "SFIRSIT"
```

determină afişarea pe ecran a mesajului "SFIRSIT"

e)
```
10 FOR i=1 TO 1 STEP 2
20 PRINT i
30 NEXT i: PRINT "SFIRSIT"
```

respectiv

```
10 FOR i=1 TO 1 STEP -2
20 PRINT i
30 NEXT i : PRINT "SFIRSIT"
```

determină rezultatul

```
1
SFIRSIT
```

• P2.

Programul realizează afişarea numerelor întregi de la 1 la 10 în ordine inversă.

```
 5 LET a$ = " "
10 FOR n=10 TO 1 STEP -1
20 PRINT n; a$;: NEXT n
```

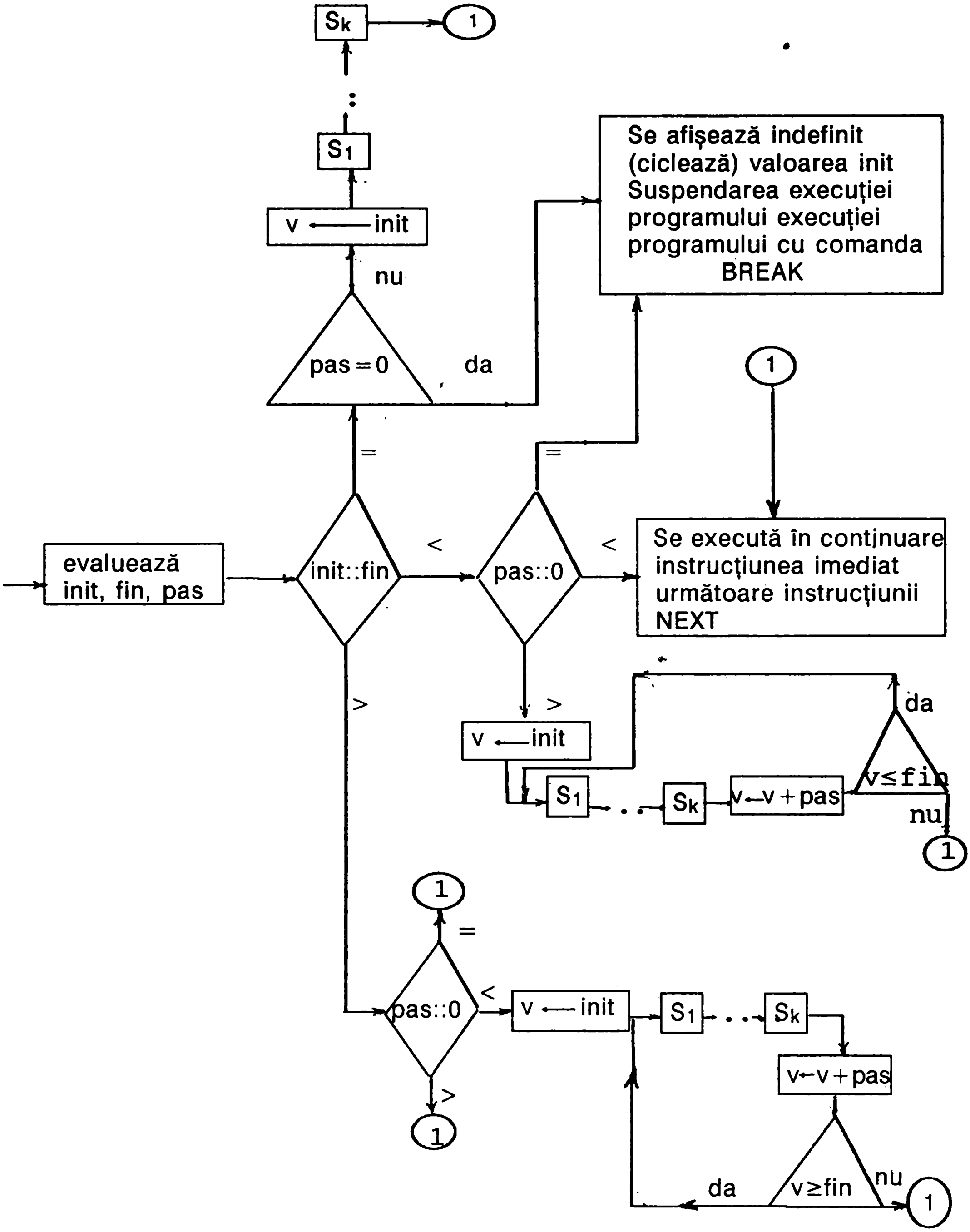
• P3.

Programul ilustrează posibilitatea ca o buclă FOR-NEXT să fie astfel încît instrucţiunea NEXT să fie plasată pe o linie de program avînd numărul de ordine mai mic decît cel corespunzător liniei de program pe care este amplasată instrucţiunea FOR asociată.

Bineînţeles, exemplul reprezintă o construcţie artificială neinteresantă din punct de vedere practic şi care are drept scop numai ilustrarea acestei posibilităţi. Programul realizează afişarea numerelor de la 1 la 10 urmată de afişarea mesajului "SFIRSIT".

```
 5 LET a$ = " ": GO TO 20
10 NEXT i : GO TO 50
20 FOR i=1 TO 10
30 PRINT i; a$;: GO TO 10
50 PRINT 1 :PRINT "SFIRSIT"
```

Pentru facilitarea înţelegerii modului în care se execută o instrucţiune FOR-NEXT vom considera următoarea diagramă în care sînt reprezentate acţiunile întreprinse în executarea acestei instrucţiuni. Convenim să reprezentăm prin S_1... S_k instrucţiunile componente ale unei bucle FOR-NEXT. Vom presupune că nici una dintre aceste instrucţiuni nu este o instrucţiune de tip GO TO.


Sk
1
S1
v ← init
nu
pas = 0
da
Se afişează indefinit (ciclează) valoarea init Suspendarea execuţiei programului execuţiei programului cu comanda BREAK
1
=
=
evaluează init, fin, pas
init::fin
<
pas::0
<
Se execută în continuare instrucţiunea imediat următoare instrucţiunii NEXT
>
>
v ← init
da
S1
Sk
v←v + pas
v≤fin
nu
1
1
=
pas::0
<
v ← init
S1
Sk
v←v + pas
>
1
da
v≥fin
nu
1

Relativ la modul în care este executată o instrucţiune FOR–NEXT este necesar de reţinut faptul că evaluarea expresiilor <expresie 1>, <expresie 2>, <expresie 3> este realizată o singură dată înainte de prima executare a grupului de instrucţiuni care alcătuiesc corpul buclei. In consecinţă, dacă în expresiile respective intervin variabile ale căror valori sînt modificate de către instrucţiunile cuprinse în corpul buclei, aceste modificări nu vor influenţa valorile pe care convenţional le-am notat cu **init, fin, pas**. De asemenea, este permisă (dar în general nejustificată) modificarea valorii variabilei de control de către instrucţiunile care compun corpul buclei.

Limbajul BASIC acceptă structuri în care buclele FOR–NEXT să fie incluse unele într–altele. Reprezentînd convenţional o buclă FOR–NEXT în modul

```
┌FOR
└NEXT
```

trebuie să respectăm ca structurile combinate din eventual mai multe bucle FOR–NEXT să fie incluse şi nu "intersectate".

```
    FOR
```

Exemplu. Structura

```
┌ FOR
│ ┌FOR
│ └NEXT
└ NEXT
```

este corectă în timp ce structura

```
 ┌FOR
┌┼FOR
│└NEXT
└ NEXT
```

este ilegală.

De asemenea, trebuie respectată condiţia ca două bucle FOR–NEXT incluse să utilizeze variabile de control distincte.

Exemple.

a)
```
 5 LET a$ = "¤"
10 LET m=1: LET n=5: LET K=1
20 FOR i=m TO n STEP k
30 PRINT "i="; i; a$; "m="; m; a$; "n="; n;a$; "k="; k
40 LET m= m+1: LET n=n+2: LET k=k+3
50 NEXT i
```

determină obţinerea următoarelor rezultate :

```
i=1 m=1 n=10 k=1
i=2 m=2 n=12 k=4
i=3 m=3 n=14 k=7
i=4 m=4 n=16 k=10
i=5 m=5 n=18 k=13
```

```
b) 5 LET a$ = "¤"
  10 LET m=1: LET n=10: LET k=1
  20 FOR i=m TO n STEP k
  30 PRINT "i="; i; a$; "m="; m; a$; "n="; n; a$; "k=";k
  40 LET m=m+1: LET n=n+2: LET k=k+3: LET i=i+2
  50 NEXT i
```

se obţin următoarele rezultate

```
i=1 m=1 n=10 k=1
i=4 m=2 n=12 k=4
i=7 m=3 n=14 k=7
i=10 m=4 n=16 k=10
```

```
c) 10 LET a=1
   20 FOR i=1 TO 5: LET a=a+i: LET b=0
   30 FOR j=1 TO a: LET b=b+j: NEXTj: PRINT b
   40 LET a=b: NEXTi
```

determină afişarea următoarelor rezultate

```
3
15
171
15400
```

4.5 Comentariu: REM

Instrucţiunea REM permite introducerea de comentarii într-un program BASIC, comentarii care sînt ignorate de către calculator, dar care permit utilizatorului simplificarea depanării eventuale a programului şi urmărirea diferitelor etape ale algoritmului.

Sintaxa instrucţiunii REM este

<instrucţiunea REM> :: = {<etichetă>} REM <comentariu>

unde <comentariu> poate fi orice secvenţă de caractere.

Exemplu.

```
a) 10 REM Acest program calculeaza produsul a
      doua matrice
b) 10 LET a=3: REM variabila a a fost initializata
```

Dacă instrucţiunea REM este prima instrucţiune dintr-o linie de program, atunci întreg conţinutul liniei de program va fi considerat drept comentariu.

Exemple.

```
10 REM abcd: PRINT 3
30 PRINT 2
```

Executarea programului determină afişarea numai a valorii 2 (instrucţiunea PRINT 3 nu este considerată de către calculator ca fiind o instrucţiune„ ci este considerată ca făcînd parte din comentariu). Acelaşi efect îl are şi includerea unei instrucţiuni REM dintr-o linie de program şi anume tot conţinutul liniei respective de program după cuvîntul cheie REM este considerat drept comentariu.

```
10 INPUT R: LET L=2*PI*R: REM Variabila L are ca
   valoare o aproximatie a lungimii cercului de raza R:
   PRINT R
20 PRINT L
```

Prin executarea acestui program va apărea afişată numai valoarea variabilei **L** nu şi valoarea variabilei **R**, deoarece instrucţiunea PRINT R este considerată componentă a comentariului a cărei prezenţă este semnalată de cuvîntul cheie REM.

4.6 Terminare: STOP, END

Instrucţiunea STOP (pentru anumite microcalculatoare END) are ca efect oprirea execuţiei programului.

Sintaxa instrucţiunii STOP este

<instrucţiunea STOP> :: = {<etichetă>} STOP

Instrucţiunea STOP se include într-un program BASIC acolo unde logica programului solicită oprirea execuţiei. Dacă programul nu utilizează subrutine, atunci este posibil ca instrucţiunea STOP să nu apară în program, execuţia programului fiind realizată automat după ce a fost executată ultima instrucţiune de program sau prin intervenţia utilizatorului (comanda BREAK). Spre deosebire de suspendarea execuţiei programului prin indicarea comenzii BREAK, în cazul în care execuţia programului se opreşte ca efect al executării unei instrucţiuni STOP, nu poate fi reluată execuţia prin utilizarea comenzii CONTINUE, ci numai printr-o relansare în execuţia dictată de comanda RUN (sau RUN n).

In general, dacă un program are subprograme atunci, devine necesară utilizarea instrucţiunii STOP ca ultimă instrucţiune a programului principal deoarece, în caz contrar, s-ar continua cu executarea instrucţiunilor componente ale subprogramelor, ceea ce în majoritatea situaţiilor contravine logicii dictate de algoritm.

4.7 Definire tablouri: DIM

Limbajul BASIC permite utilizarea structurilor de date numite uzual **tablouri** (sau **matrice**). Un **tablou** este o colecţie de variabile de acelaşi tip (numeric sau de tip string), componentele colecţiei fiind numite **variabile indexate.**

Un tablou este astfel un tip de structură de date caracterizat de un **identificator** (nume constînd dintr-o singură literă pentru tablourile de tip numeric, sau literă urmată de simbolul $ pentru tablourile de tip string), număr de dimensiuni şi conţinut. Spre deosebire de variabilele (de tip numeric sau de tip string) ale căror valori pot fi eventual nedefinite, variabilele indexate sînt întotdeauna definite şi anume calculatorul iniţializează toate componentele unui tablou cu 0 dacă tabloul este de tip numeric respectiv cu " " (stringul vid) dacă tabloul este de tip string.

Referirea la o componentă particulară a unui tablou se realizează prin indexare (unul sau doi indici după cum tabloul este de tip numeric, respectiv de tip string şi în funcţie de numărul de dimensiuni corespunzător). Informarea calculatorului asupra faptului că se intenţionează utilizarea unui tablou se realizează prin intermediul instrucţiunii DIM. Pentru fiecare tablou utilizat în cadrul unui program este necesară declararea lui prin cîte o instrucţiune DIM. O instrucţiune DIM poate fi plasată oriunde în program cu condiţia ca declararea tabloului respectiv să preceadă referirea la variabilele indexate corespunzătoare lui.

Pentru declararea unui tablou de tip numeric, instrucţiunea DIM are structura sintactică:

<instrucţiunea DIM–cazul numeric>::=

{<etichetă>} DIM <literă>

(<expresie 1>{,<expresie 2>})

unde literă reprezintă identificatorul indicat ca nume pentru tablou, iar <expresie 1>, <expresie 2> sînt expresii evaluabile (în sensul că toate variabilele componente ale acestor expresii sînt definite) şi prin ale căror evaluări rezultă date de tip numeric.

Exemple.

```
10 DIM a (100)
20 LET n=3: DIM b(10, n↑3)
```

În cazul în care instrucţiunea DIM este de tipul

DIM <literă> (<expresie 1>), se realizează declararea unui tablou de nume <literă> avînd o singură dimensiune, numărul de componente din tablou fiind dat de valoarea obţinută prin evaluarea expresiei şi aplicarea operaţiei de rotunjire rezultatului obţinut.

Dacă instrucţiunea DIM este de tipul

DIM <literă> (<expresie 1>,<expresie 2>), atunci prin această instrucţiune se

realizează declararea unui tablou cu două dimensiuni avînd numele <literă> şi avînd ca număr de componente produsul valorilor obţinute prin evaluarea expresiilor <expresie 1> şi <expresie 2> cărora li s-a aplicat operaţia de rotunjire. Ne putem reprezenta un astfel de tablou ca fiind o matrice cu **n** linii şi **p** coloane, unde **n,p** sînt valorile obţinute prin evaluările expresiilor <expresie 1>, <expresie 2>, urmate de aplicarea operaţiei de rotunjire .

Exemplu. Prin instrucţiunea DIM a (n ↑2), dacă valoarea variabilei **n** este 3 se realizează declararea tabloului unidimensional de nume **a** avînd 9 componente. Prin instrucţiunea DIM b (10, 2*3) se realizează declararea tabloului bidimensional de nume **b,** avînd 60 componente dispuse pe 10 linii şi 6 coloane.

Intr-un program BASIC este posibilă utilizarea unei litere ca nume de variabilă şi, în acelaşi timp, nume de tablou, calculatorul avînd posibilitatea să distingă între referirile la variabilă şi referirile la componentele tabloului.

Referirea la componentele unui tablou se realizează prin structuri de tipul DIM <literă> (<expresie>) sau <literă> (<expresie 1>,<expresie 2>) unde <literă> reprezintă numele tabloului, <expresie>, <expresie 1>, <expresie 2> fiind expresii prin ale căror evaluări rezultă constante de tip numeric.

În cazul unui tablou declarat prin instrucţiunea DIM ca fiind unidimensional referirea la componentele lui se realizează prin structuri de tipul <literă> (<expresie>), iar referirea la componentele unui tablou declarat prin instrucţiunea DIM ca fiind bidimensional se realizează prin structuri de tipul <literă> (<expresie 1>,<expresie 2>). O astfel de referire are ca efect evaluarea expresiei (expresiilor) indicate, aplicarea operaţiei de rotunjire rezultatului (rezultatelor) obţinute şi identificarea variabilei indexate corespunzătoare rangurilor astfel determinate.

Exemplu.

```
10 DIM a(100): LET b=3: LET a(b↑2)=5: PRINT a (b↑2)
20 FOR i=1 TO 100: INPUT a(i): NEXT i
30 DIM b(b↑3,b↑5): LET b (3,b↑2)=2: PRINT b (3,b↑2)
```

Convenţional numim **indici** expresiile <expresie>, <expresie 1>, <expresie 2> utilizate în referirea la componentele unui tablou. Dacă valoarea pentru un indice rezultată prin evaluarea expresiei urmată de aplicarea operaţiei de rotunjire este fie negativă, fie zero, fie mai mare decît valoarea declarată în instrucţiunea DIM pentru coordonata respectivă, atunci calculatorul va afişa un mesaj de eroare: "Subscript wrong".

Exemplu. Executarea programului

```
10 DIM A (10): LET a (11)=3
```

ca şi executarea programului

```
10 DIM a (10): LET a (0) = 3
```

vor determina afişarea mesajului "Subscrip wrong".

Este interesant de reţinut faptul că dimensionările sînt "dinamice" în sensul că o instrucţiune DIM ce utilizează expresii pentru indicarea numărului de componente pentru un tablou consideră pentru variabilele care apar în expresia respectivă valorile curente corespunzătoare în momentul în care instrucţiunea DIM este executată. Pentru ilustrarea acestei particularităţi vom considera următorul **exemplu**:

Prin executarea programului

```
10 LET a=3: LET b=2
20 DIM b (a↑2)
30 LET a =5
40 PRINT b(15)
```

se obţine mesajul de eroare "Subscript wrong" deoarece tabloul **b** a fost declarat în linia de program 20 ca fiind un tablou cu 9 componente (valoarea curentă corespunzătoare variabilei **a** în momentul executării instrucţiunii DIM fiind egală cu 3).

Executarea programului

```
10 LET a=3
20 DIM b (a)
30 LET b(6)=3: PRINT b(6)
```

va determina, de asemenea, afişarea aceluiaşi mesaj de eroare.

Prin executarea programului

```
 5 LET a=3
10 DIM b (a)
15 IF a<5 THEN LET a=6: GO TO 10
20 LET b(6)=13: PRINT b(6)
```

va rezulta afişarea valorii 13.

Limbajul BASIC acceptă, de asemenea, redeclarări explicite asupra aceluiaşi identificator utilizat ca nume de tablou. Desigur, aceste facilităţi trebuie exploatate de către utilizator cu multă precauţie, deoarece ele asigură o programare "comodă", dar în acelaşi timp, dacă gestiunea acestor structuri este improprie pot fi surse de erori.

Exemplu.

```
10 DIM a(5)
20 FOR i=1 TO 5: INPUT a (i): NEXT i
20 PRINT "vectorul citit": PRINT
30 FOR i=1 TO 5: PRINT a(i); "¤" ; : NEXTi:
   PRINT AT 5,0
40 DIM a (3,5)
50 FOR i=1 TO 3: FOR j=1 TO 5: INPUT a(i,j):
   NEXT j: NEXT i
55 PRINT "Matricea citita": PRINT
```

```
60 FOR i=1 TO 3: FOR j=1 TO 5:
   PRINT a(i,j); "¤" ;:
   NEXT j: PRINT: NEXT i
```

Ca aplicaţie la instrucţiunile prezentate, recomandăm exemplele:

Suma şi produsul a două marice şi bubble sort.

Declararea tablourilor de tip string se realizează prin instrucţiunea DIM avînd sintaxa

<instrucţiunea DIM–tablouri de tip string> ::=

{<etichetă>} DIM <literă> $ (<expresie 1>{,<expresie 2>})

unde <literă> reprezintă identificatorul indicat ca nume pentru tablou, iar <expresie 1>,

<expresie 2> au aceleaşi semnificaţii ca şi în cazul instrucţiunii DIM utilizate pentru declararea tablourilor de tip numeric.

Spre deosebire de tablourile de tip numeric, într-un program BASIC nu poate fi utilizat acelaşi identificator ca nume al unei variabile de tip string şi ca nume pentru tablou.

Fiecare componentă a unui tablou de tip string poate avea drept conţinut o constantă string constînd dintr-un singur caracter.

Observaţie. Conţinutul unei componente a unui tablou de tip string este o referinţă la adresa unde este memorat caracterul respectiv. Din punctul de vedere al utilizatorului, pentru moment, se poate accepta însă şi reprezentarea ca o componentă a unui tablou "conţine" un anumit caracter (eventual stringul vid " "). Componentele unui tablou string sînt iniţializate automat de către calculator cu constanta string " " (deci "valoarea" este întotdeauna definită).

În cazul tablourilor de tip string unidimensionale calculatorul acceptă identificatorul utilizat ca nume pentru tablou şi ca variabilă de tip string, efectul declarării prin DIM constînd pe de o parte în iniţializarea componentelor cu stringul vid şi pe de altă parte în restricţionarea lungimii constantei care poate fi atribuită variabilei respective în sensul că o constantă string poate fi atribuită variabilei string declarată prin instrucţiunea DIM <literă>$ (<expresie>) ca fiind un tablou string, numai dacă numărul de caractere nu depăşeşte valoarea obţinută prin evaluarea expresiei <expresie> (cu aplicarea operaţiei de rotunjire).

Exemplu.

```
10 DIM a$ (10): INPUT a$: PRINT a$
20 LET a$ (3) = "a": PRINT a$ (3)
```

Atribuirea unei constante string unei variabile declarate ca fiind şi tablou de tip string realizează asocierea pentru fiecare componentă a tabloului cîte un caracter din secvenţa care compune constanta respectivă. Referirea la componentele unui tablou de tip string unidimensional poate fi realizată atît prin utilizarea unui indice

(ca în cazul tablourilor unidimensionale de tip numeric), dar şi utilizînd funcţia TO (slice).

Exemple. 1) Prin executarea programului

```
10 PRINT a$ + "a"
```

se obţine mesajul de eroare "2 Variable not found" deoarece variabila a$ este nedefinită (nu are asociată nici o valoare).

2) Prin executarea programului

```
10 DIM a$ (1): PRINT a$ + "a"
```

se obţine afişarea caracterului **a**, variabila a$ a fost iniţializată prin "" datorită declarării DIM a$ (1). In 1) variabila a$ este nedefinită, iar în 2) variabila a$ a fost iniţializată cu stringul vid ca efect al declarării DIM a$(1).

3) Prin executarea programului

```
10 DIM a$(3): LET a$ = "abcd": PRINT a$
```

se obţine ca rezultat **abc** ceea ce se explică prin faptul că lungimea constantei string ce poate fi atribuită variabilei a$ este restricţionată de declararea DIM a$(3) la 3, tentativa de atribuire a unei constante de lungime superioară lui 3 determinînd reţinerea numai a primelor 3 caractere (de la stînga la dreapta) din constanta respectivă.

4) Prin executarea programului

```
10 DIM a$ (5): LET a$ = "abc": PRINT a$
```

se obţine **abc**, ceea ce indică faptul că este posibilă atribuirea unei constante de lungime inferioară celei declarate prin DIM . Intr-o astfel de situaţie asocierile dintre componentele tabloului şi caracterele din secvenţă se realizează de la stînga la dreapta.

Intr-adevăr, prin executarea programului

```
10 DIM a$ (5): LET a$ = "abc": PRINT a$ (2)
```

se va obţine afişarea caracterului **b**.

5) Prin executarea programului

```
10 DIM a$ (10): LET a$ (2 TO 5)= "abc":
   PRINT a$ (3): PRINT a$ (2 TO 5): PRINT a$
```

se afişează informaţia

```
b
abc
abc
```

Pentru tablourile de tip string bidimensionale declarate prin instrucțiuni de tipul DIM <literă> $ (<expresie 1>, <expresie 2>), sînt valabile toate precizările formulate pentru tablourile bidimensionale de tip numeric. Dacă **n,p** sînt valorile obținute prin evaluarea expresiilor <expresie 1> respectiv <expresie 2> (cu aplicarea operației de rotunjire), atunci ne putem imagina un astfel de tablou ca fiind o matrice cu **n** linii și **p** coloane, fiecare componentă avînd drept conținut cîte o constantă string constînd dintr-un singur caracter.

Exemplu.
```
10 DIM a$ (5,10): LET a$ (2,3) = "a": PRINT a$ (2,3)
```

In afară de referirea la componentele unui tablou de tip string bidimensional prin utilizarea indicilor (ca în cazul tablourilor de tip numeric bidimensionale) sînt posibile și alte modalități de referire specifice acestui tip de date, pe care le vom prezenta în continuare.

Calculatorul poate interpreta un tablou bidimensional de tip string și ca fiind o colecție de **n** variabile de tip string, în fiecare variabilă putînd fi înregistrată cîte o constantă string compusă din cel mult **p** caractere (unde **n** și **p** au semnificațiile precizate mai sus). Cu alte cuvinte, fiecare linie a unui tablou bidimensional de tip string poate fi referită ca fiind o variabilă de tip string în care poate fi înregistrată o informație constînd dintr-o secvență cu cel mult **p** caractere (cîte un caracter pentru fiecare componentă de pe linia respectivă). Această particularitate oferă multă flexibilitate în operarea cu secvențe de caractere în limbajul BASIC.

Exemple. 1) Prin executarea programului

```
10 DIM a$ (5,10): LET a$(2) = "abc". LET b$="¤¤¤"
20 PRINT a$(2); b$; a$(2,TO 3); b$; a$(2) (TO 3)
30 LET a$ (3,2)="A": PRINT a$ (3,2); b$; a$(3)(2 TO 2);
   b$; a$(3,2 TO 2); b$; a$ (3)(2)
40 LET a$(4) (5 TO 7)="123":
   PRINT a$(4,5 TO 7); b$; a$ (4)(5 TO 7)
```

se obține pe ecran
```
abc                    abc        abc
A       A      A       A
123          123
```

Se observă că referirile a$ (3,2), a$(3) (2 TO 2), a$(3,2 TO 2), a$ (3)(2) sînt echivalente. Analog, pentru referirile a$(4)(5 TO 7) și a$(4,5 TO 7).

Variabila b$ a fost utilizată în listele instrucțiunilor PRINT pentru inserarea unui număr convenabil de spații libere pe ecran.

● P4

Se numește **palindrom** o secvență de caractere $x_1 \ldots x_n$ cu proprietatea că

$x_i = x_{n-i+1}$ pentru toți $1 \le i \le n$.

Exemple de palindroame. 'cuc', 'cojoc', '123454321' etc.

```
 5 REM Verifica daca o secventa de caractere
   este palindrom
 7 LET b$ = " "
10 INPUT "Indicati secventa", a$
20 FOR i=1 TO INT (LEN a$/2):
   IF a$(i) <> a$ (LENa$ - i+1) THEN GO TO 40
30 NEXT i: PRINT "Secventa"; b$+a$+b$; "este
   palindrom": GO TO 50
40 PRINT "Secventa"; b$+a$+b$; "nu este palindrom"
50 INPUT "Doriti reluarea programului pentru un alt
   cuvînt? (da/nu)"; a$
60 IF a$ = "da" THEN CLS: GO TO 10
70 PRINT FLASH 1: AT 10, 10; "SFIRSIT"
```

• P5

Programul realizează numărarea aparițiilor unui anumit caracter într-o secvență de caractere dată.

```
 5 REM Numara aparitiile unui caracter într-o
   secventa data
 7 LET c$ = "¤¤¤"
10 INPUT  Indicati secventa", a$
15 INPUT "Indicati caracterul", b$
20 LET k=0: FOR i=1 TO LEN a$:
   IF a$ (i)= b$ THEN LET k=k+1
30 NEXT i: PRINT "Numarul aparitiilor caracterului";
   c$+b$+c$;"in secventa"; c$+a$+c$;
   "este egal cu";c$;k
40 INPUT "Doriti reluarea programului?(da/nu)"; k$:
   IF k$= "da" THEN CLS: GO TO 10
50 PRINT AT 10 10; FLASH 1; "LA REVEDERE"
```

• P6

Programul verifică dacă o secvență dată a$ este subsecvență a unei secvențe b$

```
 5 REM Subsecventa
 7 LET c$ = " "
10 INPUT "Indicati secventa b$", b$
15 INPUT "Indicati secventa a$", a$: LET k$="nu"
20 FOR i=1 TO LEN b$ - LEN a$ + 1
30 FOR j=1 TO LEN a$:
   IF a$(j)<>b$ (i+j-1) THEN GO TO 50
40 NEXT j: LET k$ = c$ : GO TO 60
50 NEXT i
60 PRINT "Secventa"; c$+a$+c$+k$;
   "este o subsecventa a secventei"; C$+b$
70 INPUT "Doriti reluarea programului?(da/nu)"; d$
80 IF d$ = "da" THEN CLS: GO TO 10
```

```
90 PRINT AT 10,10; FLASH 1; "SFIRSIT"
```

Observaţie. În cadrul programelor prezentate au fost utilizate instrucţiunea CLS şi opţiunea FLASH 1. Semnificaţia acestora va fi prezentată în cadrul capitolului 7.

4.8 Citire din bloc de date: READ, DATA, RESTORE

În cazul exemplelor precedente, datele care urmează să fie prelucrate de către un program au fost introduse în faza de execuţie a programului prin executarea instrucţiunilor INPUT.

În anumite situaţii, acest mod de introducere a datelor se dovedeşte a fi incomod, în special atunci cînd se doreşte repetarea execuţiei programului astfel încît anumite informaţii să rămînă nemodificate (de exemplu, valorile componentelor unor tablouri etc.)O relansare în execuţie a programului ar presupune introducerea încă o dată a tuturor datelor de intrare. Limbajul BASIC oferă posibilitatea creerii unui bloc de date, informaţia putînd fi preluată dintr-un astfel de bloc utilizînd instrucţiunea READ.

DATA

Crearea blocului de date se realizează prin utilizarea uneia sau mai multor instrucţiuni DATA.

Sintaxa instrucţiunii DATA este

<instrucţiunea DATA>:: ={ <etichetă>} DATA <listă>

unde <listă> este o secvenţă de expresii, componentele secvenţei fiind separate prin ",".

Exemplu. 10 DATA 10, "abc"+a$, a(3,2)*b+3

În cadrul unui program BASIC pot exista mai multe instrucţiuni DATA, efectul fiind cumulativ în sensul că expresiile componente ale unei instrucţiuni DATA completează blocul de date creat.

Este important de reţinut că preluarea unei informaţii din blocul de date presupune evaluarea expresiei respective, evaluarea realizată pe baza valorilor curente corespunzătoare variabilelor la momentul la care se realizează accesul la blocul de date.

Preluarea informaţiei din blocul de date este nedestructivă, existînd posibilitatea ca blocul să fie parcurs de mai multe ori în timpul execuţiei programului. Eticheta unei instrucţiuni DATA este un parametru opţional în sensul că o instrucţiune DATA poate fi inserată oriunde într-o linie de program.

READ

Instrucţiunea READ este utilizată pentru preluarea informaţiei din blocul de date. Sintaxa instrucţiunii READ este:

<instrucţiunea READ>::={ <etichetă>} READ <listă>

unde <listă> reprezintă o secvenţă de identificatori de variabile (simple sau indexate) separate prin ",".

Exemple.

```
10 READ a, b$, c, c$ (1 TO 3)
10 LET a=3: READ x,y: LET z=x+y
```

Executarea unei instrucţiuni READ constă în preluarea informaţiei necesare din blocul de date (cîte o expresie pentru fiecare componentă din lista READ), evaluarea expresiilor şi atribuirea valorilor rezultate ca valori variabilelor din lista READ. Este interesant de reţinut că în momentul executării unei instrucţiuni READ nu este neapărat necesar ca blocul de date să fi fost deja explicit creat, în sensul că nu este necesar ca instrucţiunile DATA să preceadă instrucţiunile READ.

Exemplu. Prin executarea programului

```
10 READ a,b,c : PRINT a,b,c
20 DATA 1,2,3,4,5,6
```

vor apare afişate valorile 1,2 şi 3.

Blocul de date este explorat de către calculator prin utilizarea unui pointer (element de control al poziţiei) care înainte de executarea primei instrucţiuni READ indică prima componentă a blocului de date, executarea unei instrucţiuni READ determinînd avansarea pointerului pe măsură ce informaţia este preluată din bloc, astfel încît la terminarea execuţiei instrucţiunii pointerul va indica poziţia de unde va începe preluarea de informaţie dictată de următoarea instrucţiunea READ. Tentativa de depăşire a zonei în care a fost depusă informaţia într-un bloc de date va fi semnalată de către calculator ca eroare.

Exemple. A) Executarea programului

```
10 DATA 1,2: READ a,b,c
```

va determina afişarea mesajului de eroare "E Out of DATA"

B) Prin executarea fiecăruia dintre programele

```
10 LET a=2: LET b$ = "     "
20 DATA a, a↑2, a↑3, a↑4, a↑5
30 FOR i=1 TO 5: READ x:PRINT x; b$,: NEXT i
```

respectiv

```
10 LET a=2: LET b$ = " "
20 FOR i = 1 TO 5: READ x: PRINT x; b$; : NEXT i
30 DATA a, a↑2, a↑3, a↑4, a↑5
```

se obține ca rezultat afișarea numerelor

```
2 4 8 16 32
```

ceea ce pune în evidență faptul că nu este necesar ca instrucțiunea DATA să preceadă instrucțiunile READ ce accesează blocul de date.

C) Executarea programului

```
 5 LET b$ = " "
10 FOR a= 1 TO 10 STEP 2
20 READ x: PRINT x; b$; : NEXT a
30 DATA a,a,a,a,a
```

determină afișarea numerelor

```
1 3 5 7 9
```

se observă că preluarea informației din blocul de date se realizează evaluînd expresiile aritmetice respective pe baza valorilor curente corespunzătoare variabilelor.

D) Prin executarea programului

```
10 LET a=3 : LET b=5: LET c$= "abcd":
   PRINT AT 3,3; "a="; a; AT 3,8; "b="; b;
   AT 3,20;"c$="; c$: DATA a,b,c$, a*b+2, c$+"123"
20 READ n,m,p$: PRINT AT 5,3; "n="; n; AT 5,8;
   "m="; m; AT 5,20; "p$="; p$
25 DATA 1,2,"12": READ a,c$,m,n,p$
30 PRINT AT 7,3; "a="; a; TAB8; "c$="; c$;
   TAB 20; "m="; m; AT 9,3; "n="; n; TAB20; "p$="; p$
```

se obține afișată pe ecran informația

```
a=3      b=5          c$=abcd

n=3      m=5          p$=abcd

a=17     c$=abcd123      m=1

n=2                   p$=12
```

Executarea instrucțiunilor READ dintr-un program determină explorarea blocului de date, pointerul avansînd cu cîte o poziție pentru fiecare componentă preluată (prin a cărei evaluare rezultă o valoare atribuită unei variabile din lista READ).

RESTORE

Instrucțiunea RESTORE permite deplasarea pointerului către diferite poziții în blocul de date (eventual la începutul blocului). RESTORE are structura sintactică

<instrucțiunea RESTORE> ::= {<etichetă>} RESTORE <expresie>

unde <expresie> este o expresie BASIC prin a cărei evaluare se obține o constantă tip numeric.

Executarea unei instrucţiuni de tipul RESTORE <expresie> determină deplasarea pointerului în dreptul primei poziţii care urmează sfîrşitului porţiunii din blocul de date create prin instrucţiuni DATA plasate pe linii de program, avînd numere de ordine asociate mai mici decît valoarea obţinută prin evaluarea expresiei <expresie> căreia îi este aplicată operaţia de rotunjire (deci indicînd prima poziţie din blocul de date în care urmează a fi depusă informaţia primei instrucţiuni DATA plasată pe o linie de program de etichetă mai mare sau egală decît valoarea expresiei <expresie>).

Parametrul <expresie> este opţional, în cazul în care lipseşte, maşina presupune pentru <expresie> valoarea 1.

Exemple.

```
10 DATA 1,2,3,4
20 DATA 5,6,7,8: RESTORE 15: READ a,b,c:
   PRINT a; b; c
```

conduce la afişarea rezultatului

```
567
```

```
10 DATA 1,2,3,4
20 DATA 5,6,7,8: RESTORE 20: READ a,b,c: PRINT a;b;c;
```

se obţine rezultatul

```
567
```

```
10 DATA 1,2,3,4
20 DATA 5,6,7,8: RESTORE 10: READ a,b,c,d,e,f:
   PRINT a;b;c;d;e;f;
30 RESTORE 15: READ a,b,c,d: PRINT a;b;c;d
```

determină afişarea rezultatelor

```
123456
5678
```

```
15 DATA 1
20 DATA 2: FOR i=15 TO 20: RESTORE i:
   PRINT "i="; i, : READ a: PRINT "a="; a: NEXT i
```

determină obţinerea rezultatelor

```
   i=15  a=1
   i=16  a=2
   i=17  a=2
   i=18  a=2
   i=19  a=2
   i=20  a=2
```

```
 5 DATA 1,2,3,4,5
10 READ y,z,: PRINT "y="; y, "z="; z
15 READ x: PRINT "x="; x
```

```
20 RESTORE 25: READ a,b,c: PRINT "a=";a,"b=";b,"c=";c
30 DATA 6,7,8,9,10
```

va determina afişarea următoarelor rezultate:

```
y=1       z=2
x=3
a=6       b=7
c=8
```

În cazul în care în loc de RESTORE 25 este considerată instrucţiunea RESTORE, rezultatele vor fi

```
y=1       z=2
x=3
a=1       b=2
c=3
```

Următorul program va afişa valorile unui "trasor" care indică instrucţiunea PRINT prin a cărei executare a fost realizată afişarea respectivă.

```
  5 LET s$ = " "
 10 READ a,b,c: PRINT a; s$; b; s$; c; "1":
    DATA 1,2,3,4
 20 LET a=10: LET b=20: LET c=30:
    PRINT a; s$; b; s$; c,"2":
    RESTORE: READ: a,b,c: PRINT a; s$; b; s$; c,"3"
 30 DATA 5,6,7,8,9
 40 READ a,b,c: PRINT a; s$; b; s$; c, "4"
 50 RESTORE 25: READ a,b,c: PRINT a; s$; b; s$; c,"5"
 70 DATA 100, 200, 300, 400, 500, 600, 700
    FOR i=1 TO 5: READ r: PRINT "r="; r: NEXT i
 90 READ a,b,c: PRINT a; s$; b; s$; c, "6"
100 RESTORE 60: READ a,b,c: PRINT a; s$; b; s$; c, "7"
```

Rezultatele obţinute prin executarea acestui program vor fi

```
    1         2        3        1
   10        20       30        2
    1         2        3        3
    4         5        6        4
    5         6        7        5
    n=8
    n=9
    n=100
    n=200
    n=300
  400       500      600        6
  100       200      300        7
```

```
  5 DATA 1,2,3,4,5
 10 FOR i= 1 TO 5: READ x: PRINT x; " "; :
    RESTORE i*10 : NEXT i
```

```
30 DATA 6,7,8,9,10
60 DATA 11, 12, 13, 14, 15
```

determină obţinerea rezultatelor

```
1 6 6 6 11
```

4.9 Intreruperi execuţie: PAUSE

Instrucţiunea PAUSE permite oprirea continuării execuţiei unui program pe o durată definită sau nedefinită. PAUSE poate fi utilizată atît ca o comandă cît şi ca instrucţiune într-un program.

Sintaxa instrucţiunii PAUSE este

<instrucţiunea PAUSE> ::= { <etichetă> } PAUSE <expresie>

unde <expresie> este o expresie prin a cărei evaluare (cu aplicarea operaţiei de rotunjire rezultatului obţinut) se obţine constantă de tip numeric. Dacă n este valoarea astfel obţinută, atunci trebuie respectată condiţia ca $0 \leq n \leq 65535$.

Exemplu. PAUSE 100, PAUSE 0

Efectul executării unei instrucţiuni PAUSE constă în oprirea continuării execuţiei programului pe o durată de n/50 secunde. Executarea unei instrucţiuni PAUSE poate fi întreruptă prin acţionarea oricărei taste a claviaturii.

Utilizarea instrucţiunilor PAUSE într-un program permite utilizatorului exercitarea unui control asupra etapelor în executarea unui program. Efectul executării unei instrucţiuni de tipul PAUSE 0 este de a determina suspendarea executării instrucţiunilor din program pe o durată nedefinită, momentul reluării execuţiei programului fiind ales de către utilizator care poate determina reluarea prin acţionarea unei taste oarecare a claviaturii.

4.10 Ştergere ecran: CLS

CLS (clear screen)

Instrucţiunea CLS are structura sintactică

<instrucţiunea CLS> ::= {<etichetă>} CLS

Exemplu.

```
10 IF a$ ="nu" THEN CLS
```

CLS poate fi utilizată atît ca instrucţiune în program cît şi ca o comandă directă (independentă de program).

Efectul executării instrucţiunii (comenzii) CLS constă în ştergerea informaţiei înscrise pe ecran. In particular în cazul utilizării monitoarelor color, în urma

executării unei instrucţiuni (comenzi) CLS pentru ecran (dar nu şi pentru chenar) se obţine culoarea de fond selectată eventual prin instrucţiunea (comanda) PAPER precedentă.

Exemplu.

```
10 CLS: FOR i=1 TO 100: PRINT i; " "; :
   NEXT i: PAUSE 300: CLS: FOR i=1 TO 100 STEP 2:
   PRINT i; "¤" ; : NEXT i
```

Efectul executării acestui program constă în afişarea numerelor naturale de la la 100, informaţie care rămîne vizibilă pe ecran timp de 6 secunde după car ecranul este şters, în continuare fiind afişate numerele naturale impare de la 1 100.

4.11 Gestiune spaţiu memorie: CLEAR

CLEAR poate fi utilizată atît ca instrucţiune program cît şi ca o comandă directă (independentă de program).

Sintaxa instrucţiunii CLEAR este

<instrucţiunea CLEAR> ::= {<etichetă>} CLEAR

Ca în cazul tuturor instrucţiunilor BASIC ce pot fi utilizate şi drept comenzi, CLEAR este indicată drept comandă prin tastarea directă (fără etichetă) a cuvîntului cheie CLEAR. Utilizată ca instrucţiune într-un program ea poate fi plasată oriunde într-o linie a programului. Efectul executării unei instrucţiuni (comenzi) CLEAR este de a şterge valorile curente corespunzătoare tuturor variabilelor şi tablourilor din program, ceea ce permite utilizarea economică a spaţiului de memorie disponibil.

De asemenea, executarea instrucţiunii (comenzii) CLEAR determină, ştergerea informaţiei înscrise pe ecran (efectul CLS) şi repoziţionarea pointerului corespunzător blocului de date în dreptul primei informaţii înscrise în bloc (efectul RESTORE). După executarea unei instrucţiuni (comenzi) CLEAR poziţia pentru PLOT (a se vedea paragraful destinat facilităţilor de grafică) devine pixelul de coordonate (0,0), iar stiva pentru subprograme este vidată. Trebuie reţinut faptul că executarea unei comenzi RUN în afară de lansarea în execuţie a programului determină, în particular, şi toate acţiunile pe care le realizează executarea unei instrucţiuni (comenzi) CLEAR.

4.12 Generare numere aleatoare: RANDOMIZE

RANDOMIZE

nstrucţiunea RANDOMIZE are structura sintactică

<instrucţiunea RANDOMIZE> :: = { <etichetă> }RANDOMIZE <expresie>

de <expresie> este o expresie BASIC prin a cărei evaluare rezultă o constantă merică căreia fiindu-i aplicată operaţia de rotunjire rezultă o valoare **n** cu ≤ n ≤ 65535.

arametrul <expresie> este opţional, în cazul în care lipseşte, calculatorul onsideră valoarea 0 pentru acesta. Efectul executării unei instrucţiuni RAN-OMIZE constă în considerarea drept număr pseudoaleator curent cel de al n-lea număr din secvenţa de numere pseudoaleatoare (fixată) care pot fi obţinute utilizînd funcţia RND, astfel încît o nouă apelare a funcţiei RND va determina obţinerea celui de al n + 1–lea număr pseudoaleator din secvenţă. Executarea unei instrucţiuni de tipul RANDOMIZE 0 sau RANDOMIZE determină poziţionarea la începutul secvenţei astfel încît următoarea apelare a funcţiei RND va realiza obţinerea primului număr din secvenţă.

Exemplu.

```
10 FOR i=1 TO 100: RANDOMIZE i
20 FOR x=1 TO 5: PRINT RND; "¤"; : NEXT x
30 PRINT: NEXT i
```

Obţinerea unor numere pseudoaleatoare negative sau supraunitare poate fi realizată prin apelarea funcţiei RND drept componentă a unor expresii.

• P7

```
 5 REM Programul simuleaza aruncarea zarului
10 INPUT"Indicati de cîte ori trebuie repetata
   aruncarea zarului", a:CLS
20 PRINT "Rezultatele obtinute prin aruncarea zarului
   de"; a;"ori sint:"
30 FOR i=1 TO a: PRINT INT (RND*6)-1; "¤"; : NEXT i
40 INPUT "Doriti reluarea? (da/nu)"; a$
50 IF a$ = "da" THEN CLS: GO TO 10
60 PRINT AT 10, 10; FLASH 1; "LA REVEDERE"
```

•P8

```
 5 REM Genereaza aleator pe ecran celule albe si
   negre
10 FOR i=0 TO 21 : FOR j=0 TO 31
20 IF RND > = 0.8 THEN PRINT AT i, j; "█"
30 NEXT j: NEXT i
```

4.13 Acces memorie: POKE

POKE

Instrucţiunea (comanda) POKE permite modificarea informaţiei conţinute într-un byte al memoriei. Instrucţiunea POKE are structura sintactică

<instrucţiunea POKE> :: =

{ <etichetă> } POKE <expresie 1>, <expresie 2>

unde <expresie 1>, <expresie 2> sînt expresii BASIC prin ale căror evaluări rezultă constante de tip numeric.

Evaluarea expresiei <expresie 1> este urmată de aplicarea operaţiei de rotunjire, valoarea rezultată fiind un număr între 16384 şi 65535. Prin evaluarea expresiei <expresie 2> (urmată de aplicarea operaţiei de rotunjire) trebuie să rezulte un număr între –255 şi 255.

Exemplu. `10 POKE 23609, 255`

Valoarea obţinută prin evaluarea expresiei <expresie 1> reprezintă adresa acelui byte al memoriei al cărui conţinut se doreşte a fi modificat. Valoarea obţinută prin evaluarea expresiei <expresie 2> reprezintă informaţia care trebuie înregistrată în byte-ul a cărui adresă a fost referită.

Recomandăm, în general, ca utilizarea instrucţiunii POKE să se realizeze astfel încît instrucţiunea POKE să fie unica instrucţiune a liniei de program .

POKE poate fi dată şi ca o comandă directă, sintaxa fiind în acest caz POKE <expresie 1>, <expresie 2>, unde <expresie 1>, <expresie 2> au aceleaşi semnificaţii ca şi în cazul instrucţiunii POKE.

4.14 Definire funcţii utilizator: DEF FN

În afară de funcţiile standarad furnizate de limbajul BASIC, utilizatorul are posibilitatea să-şi definească propriile funcţii. DEF FN permite definirea de către utilizator a unei noi funcţii. Sintaxa instrucţiunii DEF FN este

<instrucţiunea DEF FN> :: =

{<etichetă>} DEF FN <nume>(<listă>) = <expresie>

Semnificaţiile parametrilor sînt:

<nume> reprezintă un identificator ce va fi considerat în continuare de către calculator ca fiind numele funcţiei a cărei definire este realizată. Identificatorul utilizat ca nume pentru o funcţie nou definită trebuie să fie diferit de numele

reprezentînd cuvinte cheie BASIC.

Dacă prin instrucțiunea DEF FN respectivă este realizată definirea unei funcții prin a cărei apelare rezultă date de tip string, atunci numele funcției trebuie să fie o literă urmată de simbolul $ (în acest caz <nume>::= <litera> $).

<lista> reprezintă o listă de identificatori nume de variabile simple separați prin ',' . Identificatorii dintr-o astfel de listă trebuie să constea din cîte o singură literă, respectiv o literă urmată de simbolul $ pentru variabilele de tip string.

Parametrul <listă> poate eventual lipsi, dar și în acest caz este obligatorie apariția parantezelor.

<expresie> este o expresie BASIC în care pot apare drept componente atît identificatorii din listă cît și constante, variabile simple sau indexate, apelări de funcții standard sau funcții utilizator.

Exemple.
```
10 DEF FN r(x,y) = SQR(x*x+y*y)
10 DEF FN a(x) = (x+a*3)/y
10 DEF FN f( ) = RND*7+ INT(x+a*0.7)
10 LET x=3: DEF FN a$ (b$,x,y)=b$(x TO y)
10 DIM b$(10,10): DEF FN a$(x,y,z)=b$(x)(y TO z)
```

Referirea (apelarea) unei funcții definite de către utilizator se realizează prin includerea drept componentă a unei expresii a unei structuri de tipul FN <nume> (<listă argumente>)

unde <nume> este numele funcției referite (identificatorul utilizat ca <nume> în instrucțiunea DEF FN corespunzătoare definirii funcției), iar <listă argumente> este o secvență de expresii separate prin "," conținînd exact atîtea componente cîte componente sînt în parametrul <listă> din instrucțiunea DEF FN prin care a fost definită funcția.

Apelarea unei funcții determină evaluarea expresiilor componente ale parametrului <listă argumente> și asocierea valorilor rezultate ca valori respectiv variabilelor din parametrul <listă> (din instrucțiunea DEF FN), asocierile fiind realizate conform cu rangul corespunzător variabilei și expresiei în cele două liste. Bineînțeles că este necesară respectarea concordanței dintre tipul variabilei din <listă> și tipul constantei obținute prin evaluarea expresiei asociate din <listă argumente>. Utilizînd valorile astfel obținute pentru variabilele din <listă> se realizează evaluarea expresiei <expresie> care figurează în instrucțiunea DEF FN. Rezultatul obținut substituie apariția apelării funcției în expresia ce a solicitat apelarea respectivă. Dacă în parametrul <expresie> din instrucțiunea DEF FN apar și alte variabile decît cele ale căror nume figurează în <lista> atunci apelarea funcției va determina evaluarea expresiei <expresie> utilizînd valorile curente ale acestor variabile.

Exemplu. Prin executarea programului
```
10 DEF FN a(x)=x+y
20 FOR i=1 TO 5:
   FOR y=1 TO 5 PRINT FN a(i); "¤" ;:
   NEXT y: PRINT: NEXT i
```

Se vor obține afișate

```
2 3 4 5  6
3 4 5 6  7
4 5 6 7  8
5 6 7 8  9
6 7 8 9 10
```

Observație. Instrucțiunile DEF FN prin care se realizează definirile unor funcții pot fi plasate oriunde într-un program, eventual apelările acestora putînd să preceadă instrucțiunile DEF FN corespunzătoare.

Exemple.

```
10 PRINT FN a(2)
20 DEF FN a(x) = FN b(y) +1
30 DEF FN b(y) = y+1
```

se afișează rezultatul 4.

```
10 READ a,b: DATA 1,2: PRINT f ( )
20 DEF FN f ( ) = a+b
```

determină afișarea valorii 3.

```
10 DEF FN f(x,y,z) = x+y+z
20 LET a= FN f(1,2,3): PRINT "a="; a:
   PRINT "f(1,2,3)="; FN f(1,2,3)
```

conduce la afișarea rezultatelor

```
    a=6
   f(1,2,3) = 6
```

```
10 DEF FN a$ (bS,x,y,) = b$(x TO y)
20 PRINT FN a$ ("abcdefgh",1,3)
```

rezultă afișată secvența abc

```
10 DEF FN a$ ( ) = b$ + c$
20 READ b$, c$: DATA "abcd", "123"
30 PRINT FN a$ ( ) + FN a$ ( )
```

determină afișarea abcd123abcd123

5. Subprograme BASIC

Frecvent sînt întîlnite situaţii în care un anumit algoritm necesită repetarea de mai multe ori a unui subalgoritm pentru realizarea diferitelor etape ale acestuia. Un program care ar reprezenta algoritmul ar presupune eventuala repetare a instrucţiunilor pe baza cărora se realizează calculul implicat de subalgoritmul respectiv în toate punctele programului unde acest subalgoritm este necesar ceea ce ar conduce la obţinerea de programe cu multe instrucţiuni. O posibilitate de evitare a acestor repetări poate fi dată de utilizarea judicioasă a instrucţiunilor GO TO şi a instrucţiunilor ce realizează reatribuiri pentru variabilele cu care se doreşte efectuarea calculului reprezentat de segmentul respectiv de program. Deşi posibil, un astfel de stil de programare este incomod solicitînd efectuarea mai multor artificii, depanarea unui astfel de program fiind o operaţie ce se dovedeşte a fi destul de laborioasă.

Limbajul BASIC oferă posibilitatea utilizării subprogramelor,un subprogram fiind un segment al programului a cărui execuţie este eventual solicitată de mai multe ori în cadrul programului.

5.1 Instructiunea GO SUB

Determinarea intrării într-un subprogram este realizată prin instrucţiunea GO SUB, revenirea la instrucţiunea imediat următoare instrucţiunii GO SUB care a determinat transferul controlului către subprogram fiind realizată prin RETURN. Un subprogram BASIC apare ca segment al programului care-l utilizează, toate variabilele programului avînd un caracter global relativ la subprogramul respectiv.

GO SUB

Sintaxa instrucţiunii GO SUB este

<instrucţiunea GO SUB> ::= {<etichetă>} GO SUB <expresie>

unde <expresie> este o expresie BASIC prin a cărei evaluare rezultă o constantă de tip numeric.

Evaluarea expresiei indicate într-o instrucţiune GO SUB este realizată cu aplicarea operaţiei de rotunjire. Efectul executării unei instrucţiuni GO SUB constă în identificarea liniei de program avînd numărul de ordine **n**, unde **n** este cel mai mic număr de linie program mai mare sau egal decît valoarea obţinută prin evaluarea expresiei şi executarea în continuare a instrucţiunilor începînd cu prima instrucţiune de pe linia de program **n**.

5.2 Instructiunea RETURN

Structura sintactică a instrucțiunii RETURN este

<instrucțiunea RETURN> ::= {<etichetă>} RETURN

Efectul executării unei instrucțiuni RETURN constă în revenirea la instrucțiunea imediat următoare instrucțiunii GO SUB care a determinat intrarea respectivă în subprogram.

Exemple.

```
10 LET a=3 : GO SUB 33.2: PRINT "a=" ; a
20 STOP
33 PRINT "1" ; "¤";
34 PRINT "2"
35 RETURN
```

determină afișarea rezultatelor

```
 1 2
 a=3
```

```
10 LET a=3: GO SUB 33.6 : PRINT "a="; a
20 STOP
33 PRINT "1"; "¤" ;
34 PRINT "2"
35 RETURN
```

se obține pe ecran

```
  2
  a=3
```

```
10 LET a=3: GO SUB 33.5: PRINT "a="; a
20 STOP
33 PRINT "1"; "¤";
34 PRINT "2"
35 RETURN
```

determină afișarea rezultatelor

```
   2
   a=3
```

Delimitarea între program și eventualele sale subprograme trebuie să existe din punct de vedere logic. Dacă o astfel de delimitare nu există, atunci se va continua cu executarea instrucțiunilor ce compun segmentele gîndite ca fiind subprograme, pînă la executarea primei instrucțiuni RETURN fapt care determină afișarea unui mesaj de eroare (deoarece executarea instrucțiunii RETURN nu a fost precedată de executarea unei instrucțiuni GO SUB).

Delimitarea dintre program și segmentele corespunzătoare subprogramelor sale se realizează prin utilizarea fie a instrucțiunii STOP (respectiv END) ca ultimă

instrucţiune din programul propriu-zis, fie prin utilizarea unei instrucţiuni GO TO care determină reluarea executării programului, oprirea execuţiei putînd fi dictată de către utilizator, prin comanda BREAK.

Exemplu.

```
10 LET a=3: GO SUB 32: PRINT "a="; a
33 PRINT "1" ; "¤" ;
34 PRINT "2"
35 RETURN
```

va conduce la afişarea informaţiei

```
1  2
a = 3
1  2
```

şi a mesajului de eroare "7 RETURN without GO SUB", deoarece datorită absenţei instrucţiunii STOP, programul execută în continuare şi instrucţiunile care alcătuiesc subprogramul .

Într-un program BASIC pot fi considerate mai multe subprograme, care pot fi eventual incluse unele într-altele. In cazul executării unei instrucţiuni RETURN (pentru programele cu apelări de subprograme din subprograme sau cu structuri complexe de includere) revenirea se realizează întotdeauna la instrucţiunea imediat următoare ultimei instrucţiuni GO SUB executate. Gestiunea apelărilor subprogramelor este realizată de calculator prin crearea unei stive în care sînt reţinute numerele de linie corespunzătoare instrucţiunilor GO SUB (pe măsură ce acestea sînt executate) astfel încît numărul de linie corespunzător ultimei instrucţiuni GO SUB executate este plasat în vîrful stivei. Executarea unei instrucţiuni RETURN se realizează astfel prin preluarea din vîrful stivei a numărului de ordine corespunzător liniei la care se realizează revenirea şi ştergerea apoi a acestei informaţii din vîrful stivei (se coboară o poziţie în stivă).

Exemplu. Citiţi programul de calcul al puterii **m** a unei matrice (cap. 9).

6. GRAFICA

6.1 Noţiuni generale: UŞR, OVER

Reamintim că în memoria calculatorului un caracter este reprezentat conform cu sistemul de codificare ASCII, astfel încît referirea la un anumit caracter poate fi realizată eventual prin intermediul codului ASCII corespunzător. Setul de caractere standard împreună cu codurile ASCII corespunzătoare sînt prezentate în Anexa 1.

Funcţia CHR$ aplicată unei expresii prin a cărei evaluare rezultă o dată de tip numeric, căreia i se aplică automat operaţia de rotunjire furnizează ca rezultat al apelării caracterul (dacă există!) avînd codul ASCII egal cu valoarea obţinută în urma evaluării; de exemplu CHR$ 65 este "A".

Funcţia CODE realizează operaţia inversă celei realizate de către funcţia CHR$ şi anume funcţia CODE aplicată unei constante string constînd dintr-un singur caracter furnizează ca rezultat codul ASCII corespunzător caracterului considerat, de exemplu CODE "A" este 65.

În afară de setul de caractere standard (predefinite), utilizatorul are posibilitatea să-şi definească caracterele grafice pe care le doreşte. Orice caracter (predefinit sau creat de către utilizator) este afişat pe ecran într-o celulă, ecranul fiind partajat în 768 celule dispuse pe 24 linii şi 32 coloane.

Numerotarea liniilor şi coloanelor începe din colţul din stînga sus al ecranului, liniile fiind numerotate 0,1,..,23, respectiv coloanele 0,1,..,31. Pentru utilizator este însă disponibilă numai zona ecranului corespunzătoare liniilor de la 0 la 21, liniile 22 şi 23 fiind rezervate pentru afişarea comenzilor, mesaje de eroare etc.

Modul de lucru în care accesul la ecran se realizează prin referirea la cele 704 celule disponibile, corespunde **modului de lucru de putere rezolutivă slabă** (sau de mică rezoluţie). Modul de lucru în care accesul la ecran este realizat prin referirea la zonele elementare (pixeli) corespunde la ceea ce se numeşte **mod de lucru în rezoluţie fină** (de putere rezolutivă mare).

Fiecare celulă a ecranului este compusă din 64 zone elementare, numite **pixeli** dispuşi pe 8 linii şi 8 coloane. Pixelii componenţi ai unei aceleeaşi celule pot conţine informaţie care să corespundă fie culorii fondului fie celei asociate "cernelii" utilizate. In consecinţă, în modul de lucru de mică rezoluţie o celulă poate fi colorată în cel mult două culori.

Limbajul BASIC dispune de un set de caractere grafice predefinite, accesul la aceste simboluri fiind realizat în **modul de lucru G (graphics mode).** La majoritatea tipurilor de tastaturi caracterele grafice predefinite sînt asociate tastelor pe care sînt înscrise cifrele de la 1 la 8, codurile ASCII corespunzătoare caracterelor grafice predefinite fiind 128..143. Obţinerea caracterelor grafice corespunzătoare codurilor ASCII 128..135 se realizează prin acţionarea direct a tastei simbolului (calculatorul aflîndu-se în modul de lucru G). Pentru obţinerea caracterelor grafice corespunzătoare codurilor ASCII 136..143, este necesară acţionarea simultană a tastelor SS (Symbol Shift) şi a tastei asociate caracterului respectiv (calculatorul aflîndu-se, de asemenea, în modul de lucru G).

Prin combinarea acestor caractere pot fi realizate diferite desene pe ecran.

Exemplu. [●P9]

```
 5 REM Determina aparitia pe ecran a cuvintului BASIC
10 LET aS = "■": LET b$ = "■■" : LET c$ = "■■■" :
   PRINT AT 3,23; a$
20 PRINT AT 5,1; b$; c$; AT 5,12; a$; AT 5,18; c$; AT
   5,23; a$; AT 5,26; b$; b$
30 PRINT AT 6,2; a$; TAB 5; b$; TAB 11; c$; TAB 17;
    b$; "¤"; b$; TAB 23; a$; TAB 25; b$; TAB 29; b$
```

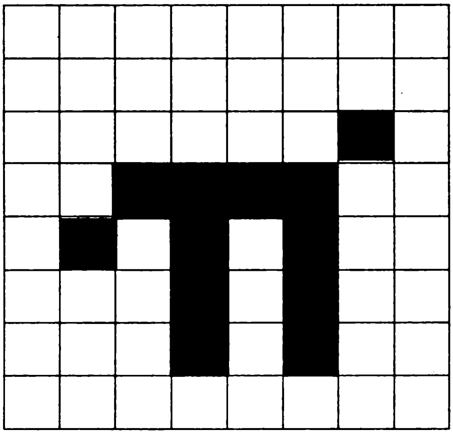
```
    b$; "¤"; b$; TAB 23; a$; TAB 25; b$; TAB 29; b$
 40 PRINT AT 7,2; a$; TAB 6; a$; TAB 10; b$;
    TAB 13; b$; TAB 17; a$; TAB 21; a$; TAB 23; a$;
    TAB 25; a$; TAB 30; a$
 50 PRINT AT 8,2; a$; TAB 5; b$; TAB 9; b$; TAB 14;
    b$; TAB 17; a$; TAB 23; a$; "¤"; a$
 60 PRINT AT 9,2; a$;"¤"; b$; TAB 9; a$; TAB 15; a$;
    "¤"; a$; TAB 23; a$; "¤"; a$
 70 PRINT AT 10,2; c$; TAB 9; a$; TAB 15; a$; "¤"; b$;
    TAB 23; a$; "¤"; a$
 80 PRINT AT 11,2; a$; "¤"; b$; TAB 9; a$; TAB 15; a$;
    TAB 18; b$; b$; "¤"; a$; "¤"; a$
 90 PRINT AT 12,2; a$; TAB 6; b$; "¤"; a$; c$; c$;
    TAB 21; a$; "¤"; a$; "¤"; a$
100 PRINT AT 13,2; a$; TAB 7; a$; "¤"; a$; TAB 15;
    a$; TAB 21; a$; "¤"; a$; "¤"; a$
110 PRINT AT 14,2; a$; TAB 6; b$; "¤"; a$; TAB 15;
    a$; TAB 21; a$; "¤"; a$; "¤"; a$
120 PRINT AT 15,2; a$; TAB 5; b$; TAB 9; a$; TAB 15;
    a$; "¤"; a$; TAB 21; a$; "¤"; a$; "¤"; a$; TAB 30; a$
130 PRINT AT 16,1; b$; c$; TAB 9; a$; TAB 15; a$; "¤";
    a$; TAB 20; b$; "¤"; a$; "¤"; b$; TAB 29; b$
140 PRINT AT 17,2; c$; TAB 9; a$; TAB 15; a$; "¤"; b$;
    b$; TAB 23; a$; TAB 26; b$; b$
```

Utilizatorul are posibilitatea să definească şi alte caractere grafice diferite de caracterele grafice predefinite. Pentru fiecare din cei 64 pixeli corespunzători unei celule în care ar fi reprezentat noul simbol grafic, trebuie indicată **culoarea** şi anume, dacă aceasta corespunde fondului sau dacă este culoarea corespunzătoare "cernelii" utilizate. Pentru un pixel pentru care se doreşte culoarea corespunzătoare fondului se va indica 0, respectiv 1 va indica opţiunea ca pixelul respectiv să fie colorat conform culorii cernelii utilizate pentru înscrierea informaţiei pe ecran.

Fiecare dintre cele 8 linii corespunzătoare unui caracter va fi astfel descrisă printr-o secvenţă de 8 simboluri 0 sau 1 indicînd culoarea asociată fiecărui pixel de pe linia respectivă, secvenţa fiind precedată de cuvîntul cheie BIN. In general cuvîntul cheie BIN este utilizat pentru a indica faptul că secvenţa pe care o precede corespunde scrierii în binar a unui număr. De exemplu BIN 00000010 sau BIN 10 vor fi interpretate de către calculator ca fiind numărul 2.

Fiecare linie a caracterului care se doreşte a fi creat, astfel descrisă, va fi memorată în cîte un byte de memorie, cei 8 bytes necesari pentru descrierea unui caracter avînd ca adrese numere consecutive. Introducerea acestei informaţii în memorie se realizează utilizînd funcţia USR şi instrucţiunea POKE. Limbajul BASIC oferă posibilitatea definirii simultan a cel mult 21 noi caractere, codurile ASCII disponibile fiind 144..164. Pentru definirea unui nou caracter grafic poate fi utilizată oricare din tastele pe care este înscrisă o literă de la A la U, accesul la tasta respectivă fiind realizat în modul de lucru G. De exemplu, să presupunem

Conform cu desenul alăturat, cele 8 linii de pixeli corespunzători vor avea respectiv reprezentările :



BIN 00000000

BIN 00000000

BIN 00000010

BIN 00111100

BIN 01010100

BIN 00010100

BIN 00010100

BIN 00000000

USR

Funcția USR realizează convertirea argumentului ce constă dintr-un singur caracter în adresa byte-ului caracterului considerat. De exemplu, USR "A" reprezintă adresa byte-ului corespunzător caracterului A (obținut în modul de lucru G). In consecință, cei 8 bytes de memorie ce vor fi utilizați pentru reprezentarea descrierii simbolului nou creat vor avea adresele USR "A", USR "A" + 1.. USR "A" + 7. Instrucțiunea POKE permite stocarea informației în memorie la adresa indicată, în cazul nostru aceasta fiind dată de funcția USR. In general, este permis ca o linie de program să conțină cel mult două instrucțiuni POKE. Recomandăm însă ca fiecare linie de program să conțină o singură instrucțiune POKE.

Pentru definirea caracterului π putem utiliza următorul program:

●P10

```
10 REM Programul creaza simbolul grafic, utilizind
   tasta corespunzatoare literei A - accesul la tasta
   fiind în modul de lucru G
20 POKE USR "A",   BIN 00000000
30 POKE USR "A"+1, BIN 00000000
40 POKE USR "A"+2, BIN 00000010
50 POKE USR "A"+3, BIN 00111100
60 POKE USR "A"+4, BIN 01010100
70 POKE USR "A"+5, BIN 00010100
80 POKE USR "A"+6, BIN 00010100
90 POKE USR "A"+7, BIN 00000000
```

După executarea acestui program o relistare a programului va pune în evidență faptul că toate aparițiile simbolului **A** accesat în modul de lucru G (inclusiv cele din program) au fost substituite cu simbolul nou creat π.

din program) au fost substituite cu simbolul nou creat π.

De exemplu, adăugînd la programul precedent liniile de program

```
100 PRINT "A":REM Simbolul A obtinut în modul de lucru C
110 PRINT "A":REM Simbolul A obtinut în modul de lucru G
120 LET A=3: PRINT A
```

o nouă lansare în execuția programului va determina afișarea

```
A
π
3
```

Este de reținut faptul că un simbol nou creat poate fi utilizat și după ce programul care a realizat creerea simbolului a fost șters din memorie (prin comanda NEW).

Simbolul nou creat poate fi utilizat în continuare pînă la o nouă resetare a calculatorului.

Observație. O particularitate a limbajului BASIC implementat pe microcalculatoare compatibile SPECTRUM este aceea că informația numerică poate fi furnizată calculatorului utilizînd baza 2.

Exemplu. Prin executarea programului

```
10 READ a
20 DATA BIN 10: PRINT "a="; a
```

va determina afișarea rezultatului a=2

Un rezultat similar se obține și în cazul executării programului

```
10 INPUT a: PRINT "a="; a
```

datele de intrare fiind introduse prin tastarea

BIN 10

De asemenea, putem utiliza informația numerică reprezentată în baza 2 și în instrucțiuni de atribuire sau în expresii componente ale unei liste PRINT.

Exemplu. Prin executarea programului

```
10 LET a = BIN 1010 : LET b= BIN 10 + a*BIN 100:
   PRINT "a="; a, "b="; b, BIN 11010
```

se vor obține rezultatele

```
a=10       b=42
26
```

Utilizînd această particularitate obținem și alte variante de program care permit definirea într-o manieră mai simplă și comodă a unui nou simbol grafic.

Exemple.

```
 5 REM Se defineste caracterul grafic π utilizind
```

```
   tasta A.
10 FOR i=0 TO 7: READ n
20 POKE USR "A" + i,n
30 NEXT i
40 DATA BIN 00000000, BIN 00000000, BIN 00000010,
   BIN 00111100, BIN 01010100, BIN 00010100,
   BIN 00010100, BIN 00000000
50 PRINT "A": REM Simbolul A obtinut prin utilizarea
   tastei corespunzatoare literei A, în modul
   de lucru G.
```

```
 5 REM Se defineste caracterul grafic π
   utilizînd tasta A.
10 FOR i=0 TO 7: READ n
20 POKE USR "A"+i, n
30 NEXT i
40 DATA BIN0, BIN0, BIN10, BIN 111100, BIN 1010100,
   BIN 10100, BIN 10100 , BIN0
50 PRINT "A": REM Simbolul A obtinut prin actionarea
   tastei A în modul de lucru G
```

Informaţia ce trebuie înregistrată la adresa indicată de funcţia USR prin instrucţiunea POKE poate fi reprezentată direct în baza 10, calculatorul realizînd în mod automat conversia în baza 2. Bineînţeles, deoarece această informaţie va fi înregistrată într-un singur byte de memorie, numerele în baza 10 ce pot fi astfel indicate trebuie să fie numere naturale **n** cu $0 \leq n \leq 255$.

De exemplu pentru definirea caracterului grafic π putem utiliza şi varianta următoare de program.

```
5 REM Se defineste caracterul grafic π utilizînd
   tasta A.
10 FOR i=0 TO 7: READ n
20 POKE USR "A"+i, n
30 NEXT i
40 DATA 0,0,2,60,87,20,20,0
```

Numerele depuse în blocul de date corespund reprezentărilor în baza 10 pentru numerele în reprezentare binară indicate în exemplele precedente. Definirea unui nou caracter grafic poate fi realizată şi într-o manieră interactivă, descrierea caracterului care se doreşte a fi creat fiind furnizată în faza de execuţie a programului.

```
 5 REM Se defineste caracterul π utilizînd tasta A
   în modul de lucru G.
10 FOR i=0 TO 7:
   INPUT ("linia "; i+1;" a caracterului"), n
```

```
30 NEXT i
```

Dacă la solicitările de introducere de date realizate de executarea instrucţiunii INPUT se tastează secvenţa de numere 0,0,2,60,87,20,20,0 (după tastarea fiecărui număr fiind acţionată tasta ENTER, atunci efectul executării acestei variante de program va consta în crearea simbolului grafic π.

In cazul utilizării variantei de programare reprezentate în exemplul 4, datele de intrare pot fi introduse şi prin tastarea secvenţei BIN0, BIN0, BIN10, BIN 111100, BIN 1010100, BIN 10100, BIN 10100, BIN0 ca răspuns al solicitării realizate de executarea instrucţiunii INPUT (după tastarea fiecărei componente a secvenţei fiind acţionată tasta ENTER.

OVER

O modalitate mai puţin utilizată pentru crearea (numai pe ecran) a unui simbol este bazată pe utilizarea opţiunii OVER 1. Opţiunea OVER 1 permite obţinerea pe ecran a unor caractere rezultate prin suprapunerea (în aceeaşi celulă) a mai multor caractere (predefinite sau create de către utilizator). Fără opţiunea OVER 1 sau dacă este formulată opţiunea OVER 0 tentativa de înscriere într-o celulă a ecranului care conţine deja un caracter se soldează cu ştergerea caracterului deja existent şi înscrierea în celula respectivă a noului caracter.

Exemplu. Prin executarea programului

```
10 PRINT AT 3,1: "abcdefghijk"; AT 3,5; "1"
```

se obţine ca rezultat

```
abcd1fghijk
```

Opţiunea OVER 1 poate fi inserată în lista instrucţiunii PRINT ceea ce va determina ca toate tentativele de înscriere a unui nou caracter într-o celulă a ecranului în care a fost deja înscris un caracter să conducă la suprapunerea celor două caractere. Trebuie reţinut faptul că efectul opţiunii OVER ca parametru într-o listă PRINT este local, numai relativ la înscrierile de informaţie determinate de instrucţiunea PRINT solicitate prin parametrii ce urmează opţiunii OVER în listă. Opţiunea OVER 0 anulează efectul opţiunii OVER 1 avînd, de asemenea, efect local.

Limbajul BASIC permite utilizarea OVER 0, OVER 1 şi ca instrucţiuni. Includerea unei instrucţiuni OVER 1 determină ca efectul de suprapunere eventuală de caractere să se realizeze pînă la executarea unei instrucţiuni OVER 0. Dacă a fost inclusă o instrucţiune OVER 1 şi într-o listă PRINT apare ca parametru opţiunea OVER 0 atunci, numai pentru înscrierile pe ecran determinate de instrucţiunea PRINT respectivă este luată în considerare opţiunea OVER 0 (opţiunea OVER 0 anulează local efectul instrucţiunii OVER 1). Analog, dacă a fost executată instrucţiunea OVER 0 şi într-o listă PRINT este formulată opţiunea OVER 1 atunci aceasta anulează local efectul instrucţiunii OVER 0.

Exemple.

```
10 PRINT AT 3,1; "abcdefgh"; OVER 1; AT 3,5;
   "1"; OVER 0; AT 3,2; "2"
```

Executarea programului conduce la afişarea rezultatului

```
a2cdefgh
```

unde simbolurile e şi 1 apar suprapuse.

```
10 PRINT AT 3,1; "abcdefgh"; OVER 1; AT 3,5; "1";
   OVER 0; AT 3,1; "2"; AT 3,2; "3"; OVER 1; AT 3,4; "4"
```

Executarea programului conduce la afişarea informaţiei 23 cefgh, unde simbolurile e şi 4 respectiv f şi 1 apar suprapuse.

```
10 OVER 1: PRINT AT 3,1; "abcdefgh";
   AT 3,5; "1"; OVER 0; AT 3,1; "2"; AT 3,3; "3"
20 PRINT AT 2,1; "ABCDEFGH"; AT 2,5; "1"
30 OVER 0: PRINT AT 4,1; "MNPQ"; AT 4,1; "1"; OVER 1;
   AT 4,2; "2"
40 PRINT AT 5,1; "12345"; AT 5,2; "A"
```

Prin executarea acestui program vor apare afişate:

```
    1
ABCDEFGH
```
– pe linia 2, simbolurile 'E' şi '1' fiind suprapuse.

```
    1
2b3defgh
```
– pe linia 3, simbolurile 'e' şi '1' fiind suprapuse.

```
 2
1NPQ
```
– pe linia 4, simbolurile 'N' şi '2' fiind suprapuse.

```
1A345
```
– pe linia 5.

6.2 Instructiunile SCREEN$, PLOT, DRAW, CIRCLE, POINT

Limbajul BASIC oferă funcţii şi instrucţiuni care asigură posibilităţi de efectuare de grafică cu calculatorul.

SCREEN$

Funcţia SCREEN$ permite obţinerea informaţiei afişate într-o celulă a ecranului.

Apelarea funcţiei SCREEN$ se realizează conform sintaxei

<funcţia SCREEN$> ::= SCREEN$ (<expresie 1>, <expresie 2>)

unde <expresie 1>, <expresie 2> sînt expresii prin ale căror evaluări rezultă constante de tip numeric. Evaluările expresiilor <expresie 1>, <expresie 2> sînt realizate cu aplicarea operației de rotunjire, valorile astfel obținute trebuind să fie între 0 și 21 pentru <expresie 1> respectiv între 0 și 31 pentru

<expresie 2>.

Rezultatul apelării funcției SCREEN$ este caracterul care se află înscris pe ecran în celula corespunzătoare liniei și coloanei de numere valorile rezultate prin evaluarea expresiilor <expresie 1> și respectiv <expresie 2>.

Exemplu.

```
10 IF SCREEN$(10,15)="*" THEN PRINT AT 10,15; "■"
```

Funcția SCREEN$ poate fi utilizată împreună cu comenzile SAVE și LOAD pentru copierea pe bandă a informației afișate pe ecran, respectiv pentru copierea de pe bandă și afișarea pe ecran a informației.

Observație. In cazul anumitor tipuri de microcalculatoare (de exemplu, microcalculatorul SPECTRUM), funcția SCREEN$ permite preluarea informației afișate pe ecran și eventual prelucrarea ei în continuare numai pentru caracterele care nu sînt caractere grafice predefinite sau create de utilizator.

Exemple

```
 5 REM Exemplu de preluare "corecta" a informatiei de
   pe ecran
10 PRINT AT 3,3; "a"
20 LET a$ = SCREEN$ (3,3): PRINT AT 4,3; "a$="; a$
30 PRINT AT 5,5; "3": PRINT AT 6,5; SCREEN$ (5,5)
40 PRINT AT 7,5; SCREEN$ (3,3); AT 7,10; SCREEN$ (5,5)
50 IF SCREEN$ (3,3) ="a" THEN PRINT "da"
60 IF SCHEEN$ (5,5) <> "B" THEN PRINT "DA"
```

Rezultatele obținute prin executarea acestui program sînt:

```
   a
   a$ = 3
     3
     3
     a    3
  da
  DA
```

```
 5 REM Exemplu de preluare "incorecta" a informatiei de
   pe ecran
10 PRINT AT 3,3, "■"
15 IF SCREEN$ (3,3) = "■" THEN PRINT "da, este ■"
16 LET a$ = SCREENS (3,3): PRINT "a$ = "; a$
17 PRINT AT 10, 10; a$
18 PRINT AT 10,15; SCREEN$ (3,3)
```

Rezultatele afișate pe ecran obținute prin executarea acestui program sînt:

■
a$ =

•P11

```
5 REM Substituirea pe ecran a literelor format mic prin
  majusculele corespunzatoare într-un text cu cel mult
  704 caractere.
10 INPUT "Introduceti textul", a$: CLS
20 FOR i=1 TO LEN a$: LET l = INT (i/32):
   LET c= i-l*32: PRINT AT l, c; a$(i): NEXT i
30 FOR i=0 TO l: FOR j=0 TO 31: LET b$ = SCREEN$ (i,j)
40 IF (CODE b$ <= 122) AND (CODE b$ >= 97) THEN
   PRINT AT i,j; CHR$ (CODE b$ - 32)
50 NEXT j : NEXT i : CL$
60 REM Copiaza informatia de pe ecran în variabila c$
   si afiseaza textul initial si textul modificat
70 FOR i=0 TO l: FOR j= 0 TO 31: LET c$
   (32*i+j+1) = SCREENS (i,j): NEXT j: NEXT i
80 PRINT AT 10,10; "TEXTUL INITIAL": PAUSE 200 :
   PRINT a$: PAUSE 300: PRINT AT 10, 10;
   "TEXTUL MODIFICAT": PAUSE 200: PRINT c$
```

Din punctul de vedere al puterii de rezoluţie accesul la ecran este posibil atît în modul de lucru în rezoluţie slabă (low resolution) cît şi în modul de lucru cu putere rezolutivă mare (high resolution). In cazul modului de lucru cu rezoluţie slabă, unitatea de informaţie este **caracterul**, şi zonele elementare corespunzătoare ecranului sînt celulele dispuse pe 24 linii şi 32 coloane. Numerotarea liniilor şi coloanelor se realizează prin numerele naturale de la 0 la 23 (pentru linii), respectiv prin numerele naturale de la 0 la 31 (pentru coloane) numerotarea începînd din colţul din stînga sus al ecranului.

Corespunzător modului de lucru cu rezoluţie fină unitatea elementară este **pixelul**, fiecare celulă a ecranului corespunzînd unei zone rectangulare compusă din 64 pixeli dispuşi pe 8 linii şi 8 coloane. In modul de lucru cu rezoluţie fină ecranul este partajat în 176x256 pixeli dispuşi pe 176 linii şi 256 coloane. Numerotarea liniilor şi a coloanelor se face prin numerele naturale 0..175 (pentru linii), respectiv 0..255 (pentru coloane), numerotarea începînd din colţul din stînga jos al ecranului. Convenţional, numărul liniei şi numărul coloanei unui anumit pixel le vom numi **coordonatele** pixelului. Astfel prin pixelul de coordonate x,y, se va înţelege pixelul avînd **abscisa x** (numărul corespunzător coloanei) şi **ordonata y** (numărul corespunzător liniei).

Pentru efectuarea de grafică în rezolţie fină, limbajul BASIC dispune de instrucţiunile PLOT, DRAW, CIRCLE şi funcţia POINT.

PLOT

Instrucţiunea PLOT are structura sintactică

< instrucţiunea PLOT > :: = { < etichetă > }PLOT < expresie 1 >, < expresie 2 >

unde <expresie 1>, <expresie 2> sînt expresii prin ale căror evaluări (cu aplicarea operației de rotunjire) rezultă constante numerice de valori între 0 și 255 pentru <expresie 1>, și între 0 și 175 pentru <expresie 2>.

Exemple.

```
PLOT 10, 20
10 LET a=7: PLOT 30, a
20 PLOT 30, 50: PLOT 10,15: LET a=7
```

Efectul executării unei instrucțiuni PLOT x,y constă în colorarea în culoarea corespunzătoare "cernelii" utilizate, a pixelului avînd coordonatele **x,y**.

Exemplu.

```
 5 REM Coloreaza aleator cite un pixel al ecranului de
   fiecare data cind este actionata tasta ENTER
10 PLOT INT (RND*256), INT(RND*176):
   INPUT a$: GO TO 10
```

•P12

```
 5 REM Traseaza o aproximatie a graficului functiei
   sinus pentru valori ale argumentului între 0 si 2
10 FOR n=0 TO 255
20 PLOT n,88 + 80*SIN (n/128*PI)
30 NEXT n
```

•P13

```
 5 REM Traseaza o aproximatie a graficului functiei SQR
   pentru valori ale argumentului intre 0 si 4.
10 FOR n=0 TO 255
20 PLOT n,80*SQR (n/64)
30 NEXT n
```

Observație. Opțiunea AT prezentată anterior oferă posibilitatea localizării unei anumite celule a ecranului (în modul de lucru cu rezoluție slabă) în care urmează să fie înscrisă informația, în timp ce instrucțiunea PLOT realizează accesul și colorarea unui pixel al ecranului (în modul de lucru cu putere rezolutivă mare).

DRAW

Instrucțiunea DRAW are structura sintactică

<instrucțiunea DRAW>::=

{<etichetă>} DRAW <expresie 1>, <expresie 2>{,<expresie 3>}

unde <expresie 1>, <expresie 2>, <expresie 3> sînt expresii prin ale căror evaluări rezultă constante de tip numeric.

Evaluările expresiilor <expresie 1>, <expresie 2>, <expresie 3> sînt urmate de aplicarea operației de rotunjire.

Exemple.

```
DRAW 10,-20
10 LET a=3: DRAW a↑3,20: PLOT 50,50: DRAW -7,0
20 DRAW 10,15: PLOT 100,100: DRAW -20,-30
```

Efectul executării unei instrucţiuni

DRAW <expresie 1>, <expresie 2>

este de a trasa în culoarea selectată pentru cerneală un segment de dreaptă avînd extremităţile pixelii de coordonate x_0, y_0 şi respectiv $x_0 + x$, $y_0 + y$, unde x,y sînt valorile rezultate prin evaluările expresiilor <expresie 1>, <expresie 2>, iar x_0, y_0 reprezintă coordonatele originii ($x_0 = 0$, $y_0 = 0$). Dacă executarea instrucţiunii DRAW este precedată de executarea unei alte instrucţiuni DRAW sau de o instrucţiune PLOT, atunci x_0, y_0 reprezintă coordonatele pixelului "atins" prin executarea instrucţiunii PLOT sau DRAW precedente. De exemplu, dacă executarea unei instrucţiuni DRAW este precedată de executarea uneia dintre instrucţiunile RUN, CLEAR, CLS, NEW atunci $x_0 = y_0 = 0$.

• P14

```
 5 REM Trasarea unui segment de dreapta avînd
   ca extremitati pixelii de coordonate (0, 100)
   respectiv (80,65).
10 PLOT 0,100: DRAW 80, -35
```

•P15

```
 5 REM Programul determina trasarea unui triunghi
10 PLOT 127, 150
20 DRAW 70, -100
30 DRAW -140, 0
40 DRAW 70,100
```

•P16

```
 5 REM Trasarea unui dreptunghi avind ca virfuri
   pixelii de coordonate (50,50), (150,50),
   (150,100), (50,100)
10 PLOT 50,50: DRAW 100,0: DRAW 0,50: DRAW -100,0
```

•P17

```
 5 REM Programul determina trasarea unei stele cu
   5 colturi
10 PLOT 128,174
20 DRAW 70,-140
30 DRAW -152, 80
40 DRAW 164, 0
50 DRAW -150, -80
60 DRAW 70,140
```

Efectul executării unei instrucţiuni DRAW de tipul DRAW <expresie 1>, <expresie 2>, <expresie 3> constă în trasarea unui arc de cerc avînd extremităţile în pixelii de coordonate x_0, y_0 respectiv $x + x_0$, $y + x_0$, valoarea rezultată prin

evaluarea expresiei <expresie 3> indicînd numărul de radiani corespunzători arcului trasat, iar x_0, y_0, x,y avînd aceleaşi semnificaţii ca şi în cazul instrucţiunii DRAW <expresie 1>, <expresie 2>.

Dacă valoarea rezultată prin evaluarea expresiei <expresie 3> este pozitivă, atunci este considerat sensul invers acelor de ceasornic (sensul trigonometric), iar în cazul unei valori negative pentru acest parametru este considerată parcurgerea în sensul acelor de ceasornic.

Exemplu. Prin executarea programului

```
    10 PLOT 100, 100: DRAW 50,50, PI
```

va rezulta pe ecran un semicerc avînd ca extremităţi pixelii de coordonate 100,100 şi 150, 150.

CIRCLE

Structura sintactică a instrucţiunii CIRCLE este

<instrucţiunea CIRCLE::=

{<etichetă>} CIRCLE <expresie 1>, <expresie 2>, <expresie 3>

unde <expresie 1>, <expresie 2>, <expresie 3> sînt expresii prin ale căror evaluări rezultă constante de tip numeric.

Evaluările expresiilor <expresie 1>, <expresie 2>, <expresie 3> se realizează cu aplicarea operaţiei de rotunjire, valorile rezultate trebuind să fie numere între 0 şi 255 pentru <expresie 1>şi între 0 şi 175 pentru <expresie 2>. Valoarea rezultată prin evaluarea expresiei <expresie 3> trebuie să fie un număr pozitiv.

Exemple.

```
CIRCLE 100, 100, 50
10 CIRCLE 100, 100, 50: LET a=3
10 LET a=5: CIRCLE 30*a, 20*a, a↑2
```

Efectul executării unei instrucţiuni CIRCLE constă în trasarea în culoarea selectată pentru "cerneală" a cercului cu centrul pixelul de coordonate **x,y** şi de rază **r**, unde **x,y** sînt valorile obţinute prin evaluarea expresiilor <expresie 1>, <expresie 2>, iar **r** este valoarea rezultată prin evaluarea lui <expresie 3>.

•P18

```
 5 REM Trasarea unei stele cu 5 colturi inscrise
   intr-un cerc
10 PLOT 128, 174
20 DRAW 70,-140: DRAW -152, 80: DRAW 164, 0
30 DRAW -150, -80: DRAW 70, 140: CIRCLE 128, 87, 87
```

Observaţie. După executarea unei instrucţiuni CIRCLE poziţia ultimului pixel rezultat prin trasarea cercului este nedefinită din punctul de vedere al utilizatorului. In consecinţă, recomandăm ca după executarea unei instrucţiuni

CIRCLE, următoarea instrucţiune DRAW să fie precedată de o instrucţiune PLOT, pentru precizarea pixelului origine a segmentului care trebuie trasat.

•P19

```
 5 REM Programul determina trasarea unui cerc utilizind
   functiile SIN si COS
10 FOR n=0 TO 2*PI STEP PI/180
20 PLOT 100+80*COSn, 87+80* SINn
30 NEXT n
40 CIRCLE 150, 87, 80
```

Prin executarea acestui program se poate observa că instrucţiunea CIRCLE se execută mult mai rapid, dar oferă o precizie mai mică.

•P20

```
  5 REM Trasarea aproximativa a graficului unei functii
    pentru valori ale argumentului în intervalul [-n,n]
    n indicat de utilizator
  6 REM Functia trebuie data sub forma unei expresii,
    variabila fiind notata cu x; expresia
    corespunzatoare functiei este inregistrata
    intr-o variabila string.
 10 PLOT 0,87: DRAW 255, 0
 20 PLOT 127,0: DRAW 0,175
 30 INPUT "n="; s
 31 INPUT "Indicati expresia functiei"; e$
 35 LET t=0
 40 FOR f=0 TO 255
 50 LET x=(f-128) * s/128: LET y = VAL e$
 60 IF ABSy>87 THEN LET t=0 : GO TO 100
 70 IF NOT t THEN PLOT f, y+88: LET t=1: GO TO 100
 80 DRAW 1, y-v
100 LET v = INT(y+.5)
110 NEXT f
```

Instrucţiunile DRAW, PLOT, CIRCLE pot fi utilizate în combinaţie cu instrucţiuni sau comenzi prin care să se asigure o anumită cromatică pe ecran (în cazul monitoarelor color).(vezi I.7)

POINT

Sintaxa apelării funcţiei POINT este

<funcţia POINT> ::= POINT (<expresie 1>,<expresie 2>)

unde <expresie 1>, <expresie 2> sînt expresii prin ale căror evaluări rezultă numere întregi nenegative din intervalul [0,255] pentru <expresie 1> respectiv din intervalul [0,175] pentru <expresie 2>.

Exemplu:

```
POINT (17,18)
```

Rezultatul apelării funcţiei POINT este reprezentat de una din valorile 0 sau 1, după cum pixelul avînd coordonatele <expresie 1>, <expresie 2> este colorat în culoarea corespunzătoare fondului, sau în culoarea "cernelii" utilizate.

Exemplu:

```
CLS: PRINT POINT (0,0): PLOT 0,0: PRINT POINT (0,0).
```

Observaţie. Funcţia POINT realizează relativ la instrucţiunea PLOT (modul de lucru de fină rezoluţie) acelaşi rol pe care-l îndeplineşte funcţia SCREEN$ relativ la PRINT (în modul de lucru de putere rezolutivă slabă).

Programul următor ilustrează posibilităţi de utilizare a funcţiei POINT; sînt generate aleator un număr de puncte pe ecran după care este generată căutarea aleatoare a acestora. De fiecare dată cînd este întîlnit unul din punctele generate este emis un anumit sunet (instrucţiunea BEEP urmează să fie prezentată) şi sînt afişate mesaje prin care se indică proporţia încercărilor soldate cu succes.

```
10 REM Test pentru utilizarea lui PLOT si POINT
20 FOR a=1 TO 100
30 PLOT RND*100, RND*100
40 NEXT a
50 LET c=0
60 FOR g=1 TO 1000
70 IF POINT (RND*100, RND*100)= 1 THEN BEEP 1,1: LET
   c=c+1: PRINT AT 1,1; "NUMAR INCERCARI REUSITE
   ="; c/g*100; "%"; AT 3,1; "DIN "; g; " INCERCARI "
80 NEXT g
```

7. CROMATICA

7.1 Controlul culorii

Microcalculatoarele compatibile cu calculatorul SPECTRUM şi care dispun de unitate de afişaj color pot utiliza o gamă de 8 culori fiecare culoare avînd asociat cîte un cod. Culorile şi codurile corespunzătoare sînt:

negru	(cod 0),	albastru	(cod 1),
roşu	(cod 2),	roz	(cod 3),
verde	(cod 4),	albastru deschis (cyan)	(cod 5),
galben	(cod 6),	alb	(cod 7).

Controlul culorii pe ecran se realizează prin indicarea codurilor corespunzătoare culorilor dorite pentru chenar, fond şi respectiv "cerneala" utilizată pentru scrierea informaţiei. De asemenea sînt posibile exercitarea controlului asupra gradului de luminozitate (**brightness**) şi formularea de opţiuni asupra afişării informaţiei cu luminozitate variabilă (efectul **flash**) în anumite zone ale ecranului. Aceste funcţiuni se realizează prin utilizarea instrucţiunilor (comenzilor): INK, PAPER, BORDER, FLASH, BRIGHT, INVERSE, OVER.

7.2 Instrucţiunile INK, PAPER, BORDER, FLASH, BRIGHT, INVERSE, OVER

INK

Instrucţiunea INK poate fi utilizată atît ca instrucţiune program cît şi ca o comandă directă adresată calculatorului .

Sintaxa instrucţiunii /comenzii INK este

<instrucţiunea INK> :: = { <etichetă> } INK <expresie>

<comanda INK> :: = INK <expresie>

unde <expresie> reprezintă o expresie BASIC din a cărei evaluare împreună cu aplicarea operaţiei de rotunjire, rezultă unul din numerele 0..7.

Anumite variante de BASIC acceptă ca valoare pentru rezultatul evaluării parametrului <expresie> şi valorile 8 şi 9, acestea avînd semnificaţii speciale. Deoarece nu toate microcalculatoarele avînd implementate variante de BASIC acceptă valorile 8,9 nu vom insista asupra lor.

Efectul executării unei instrucţiuni (comenzi) INK este de a determina ca în continuare informaţia afişată pe ecran să apară înscrisă în culoarea al cărui cod este dat de valoarea obţinută prin evaluarea parametrului <expresie> (pînă la executarea unei noi instrucţiuni/comenzi INK).

O particularitate interesantă este aceea că INK poate fi inclusă într-un program ca instrucţiune, poate fi dată ca o comandă (independentă de program), dar poate fi utilizată în combinaţie cu instrucţiunile PRINT, INPUT, PLOT, DRAW, CIRCLE, caz în care are statutul unei opţiuni.

Efectul executării unei instrucţiuni INK este global, în sensul că după executarea acestei instrucţiuni şi pînă la executarea unei noi instrucţiuni INK (sau formularea unei opţiuni INK) informaţia care urmează să fie afişată pe ecran prin executarea instrucţiunilor PRINT, PLOT, DRAW, CIRCLE va fi înscrisă în culoarea codului indicat ca valoare a parametrului <expresie>. Este interesant de remarcat faptul că după executarea unui program în care apar una sau mai multe instrucţiuni INK, solicitarea unei listări a programului va pune în evidenţă că inclusiv liniile programului vor apare înscrise cu culoarea codului indicat de ultima instrucţiune INK executată.

De exemplu, dacă ultima instrucţiune de tip INK dintr-un program este INK 7,

atunci după executarea programului (în cazul în care culoarea fondului este culoarea albă) o nouă solicitare de listare a programului va determina ca liniile programului să fie înscrise cu cerneală albă, deci devin invizibile.

Utilizatorul poate formula eventual comanda INK 0 fapt care va determina ca următoarea solicitare de listare a programului să determine afişarea liniilor de program utilizînd "cerneală" neagră (deci vizibile).

Exemplu: Prin executarea programului

```
10 LET a=3: INK 7: LET b=5
20 PRINT a: INK 0: PRINT b
30 INK 0: INPUT a,b,c: PRINT a;b;c, "1"
40 INK 7: PRINT a;b;c, "2"
```

va apare afişată informaţia

```
      5
```

după care va fi solicitată introducerea de date. Valorile indicate de către utilizator ca valori pentru **a,b** respectiv **c** vor apare în continuare afişate o singură dată, şi anume ca efect al executării instrucţiunii PRINT din linia 30. De fapt aceste valori apar afişate şi ca efect al executării instrucţiunii PRINT din linia 40 numai că pentru afişaj este utilizată culoarea albă, deci ele sînt invizibile pentru utilizator. Ne putem convinge de acest lucru substituind instrucţiunea INK 7 din linia de program 40 prin INK 2, INK 3 etc sau pur şi simplu eliminînd-o.

După executarea programului considerat nu mai este posibilă afişarea în vizibil a liniilor programului (prin comanda LIST) decît dacă în prealabil a fost dată o comandă de tipul INK 0.

Exemplu.

```
 5 REM Trasarea unei stele rosii în 5 colturi
   inscrise intr-un cerc albastru
10 INK 2: PLOT 128, 174: DRAW 70, -140
20 DRAW -152, 80: DRAW 164,0
30 DRAW -150,-80: DRAW 70, 140
40 INK 1: CIRCLE 128, 87, 87
```

INK poate fi utilizată în combinaţie cu instrucţiunile de afişaj PLOT, PRINT, DRAW, CIRCLE urmînd cuvîntului cheie corespunzător instrucţiunii, conform cu sintaxa

INK <expresie>

Exemplu:

```
60 CIRCLE INK 4; 128, 88, 87
70 PRINT INK 2; a,b
80 INK 2: PLOT 10, 10: DRAW INK 5; 20, -10
90 PRINT AT 11,16; INK 2; "*"
```

Efectul utilizării INK ca opţiune în combinaţie cu înstructiunile de afişaj este local, în sensul că numai pentru afişarea informaţiei determinată de instrucţiunea

respectivă va fi utilizată culoarea al cărui cod este indicat. După terminarea execuției instrucțiunii de afișaj se revine la culoarea "globală" existentă (în lipsa unei precizări explicite culoarea "globală" pentru cerneală este negru).

Exemplu: Prin executarea programului

```
10 INK 7: PRINT 0
20 PRINT INK 0; "1"; INK 7; "2"; INK 0; "3"
30 PRINT "abcd"
40 INK 0: PRINT "ABCD"; INK 7; "MNPQ"
50 PRINT "GATA"
```

va rezulta pe ecran

```
1¤3
ABCD
GATA
```

PAPER

Instrucțiunea PAPER permite selectarea (globală sau locală) a culorii utilizate pentru fondul ecranului.

Sintaxa instrucțiunii PAPER este:

<instrucțiunea PAPER> ::= {<etichetă>} PAPER <expresie>

unde <expresie> are aceeași semnificație ca și în cazul instrucțiunii INK. PAPER poate fi utilizată atît ca o comandă directă adresată calculatorului, cît și ca o instrucțiune într-un program și ca opțiune în combinație cu instrucțiunile de afișaj PRINT, PLOT, DRAW, CIRCLE. Efectul executării unei instrucțiuni/comenzi PAPER constă în determinarea pentru fondul ecranului a culorii al cărui cod rezultă prin evaluarea parametrului <expresie> (evaluare cu aplicarea operației de rotunjire). După executarea unui program, o nouă listare a programului va fi realizată în cromatica pentru fondul ecranului corespunzător ultimei instrucțiuni sau comenzi PAPER executată. Pentru obținerea unui fond de aceeași culoare pe tot ecranul este necesar ca în program după instrucțiunea PAPER să fie executată instrucțiunea CLS. Utilizarea combinată a instrucțiunilor de afișaj cu PAPER ca opțiune în efect local (ca și în cazul utilizării opțiunii INK) permite obținerea unor efecte cromatice speciale.

In cazul în care PAPER este utilizată ca opțiune în combinație cu instrucțiunile de afișaj, efectul este local (numai pentru afișarea informației determinată de instrucțiunea respectivă), sintaxa fiind

PAPER <expresie>

•P22.

```
 5 REM Deseneaza un triunghi "plin" colorat in rosu pe
   fond galben
10 PAPER 6: INK 2: CLS
20 FOR x=-100 TO 100
 30 PLOT 128, 150
 40 DRAW x, - 120
```

```
50 NEXT X
```

```
 5 REM Afisarea unor patrate colorate, fondul
   ecranului fiind negru
10 PAPER 0 : CLS
20 FOR x=7 TO 0 STEP-1
30 INK x
40 FOR L=11-x TO 11+x
50 FOR c = 16-x TO 16+x
60 PRINT AT L,c; "█"
70 NEXT c
80 NEXT L
90 NEXT x
```

```
  5 REM Deseneaza triunghiuri colorate in galben,
    fondul ecranului fiind albastru
 10 PAPER 1: INK 6: CLS
 20 FOR y=0 TO 20 STEP 2
 40 PLOT 0,y
 50 DRAW 255, 0
 60 NEXT y
 70 FOR n=100 TO 220 STEP 30
 80 FOR x =-10-n/10 TO 10+n/10
 90 PLOT n, 35+n/10
100 DRAW x, -n/4
110 NEXT x
120 NEXT n
```

Observație. Deoarece un pixel al ecranului este utilizat fie pentru înscrierea de informație, fie aparține fondului rezultă că fiecare pixel este colorat fie în culoarea selectată pentru cerneală, fie în culoarea selectată pentru fond. In modul de lucru cu putere de rezoluție mare avem astfel posibilitatea ca o celulă a ecranului (unitatea în cazul modului de lucru de mică rezoluție) să conțină pixeli colorați în mai mult de două culori.

Programul următor determină colorarea aleatoare a pixelilor unei celule. Pentru asigurarea unei mai bune vizualizări, programul realizează, de asemenea, reproducerea cromaticii respective "la scară mai mare".

• P23

```
 1 REM Colorarea aleatoare a pixelilor unei celule
   a ecranului
 5 DIM a(8,8)
10 FOR i=0 TO 7: FOR j=0 TO 7: LET b=RND*7
11 PLOT INK b; 120+i, 120+j; LET a(i+1,j+1)=b
12 NEXT j: NEXT i
15 PRINT AT 3,5
20 FOR i=1 TO 8: FOR j=1 TO 8: PRINT INK a(i,j);"█":
   NEXT j
30 PRINT AT 3+i,5: NEXT i
```

BORDER

Instrucţiunea/comanda BORDER permite precizarea culorii dorite de către utilizator pentru chenarul ecranului (porţiunea din ecran care delimitează fondul utilizat pentru afişajul informaţiei). BORDER poate fi utilizată ca şi PAPER şi INK, atît ca instrucţiune program, respectiv comandă directă (independentă de program), cît şi în combinaţie cu instrucţiunile de afişaj.

Sintaxa instrucţiunii BORDER este:

<instrucţiunea BORDER> :: = {<etichetă>} BORDER <expresie>

unde parametrul <expresie> are aceeaşi semnificaţie ca şi în cazul instrucţiunilor PAPER şi INK .

BORDER utilizată drept comandă directă trebuie formulată conform cu sintaxa BORDER <expresie>. Includerea opţiunii BORDER ca opţiune într-o listă corespunzătoare unei instrucţiuni de afişaj determină selectarea culorii pentru chenar numai pentru afişarea informaţiei indicate de instrucţiunea respectivă (are ca PAPER şi INK un efect local). In acest caz sintaxa ce trebuie respectată este BORDER <expresie>.

●P24

```
 5 REM Creeaza un nou caracter si afiseaza caracterul in
   toate celulele ecranului in culori generate aleator
 6 REM Caracterul  nou creat corespunde tastei "S"
10 FOR x=0 TO 7: READ y
20 POKE USR "S" + x,y: REM Accesul la tasta "S" trebuie
    realizat in modul de lucru G
30 NEXT x
40 DATA 60, 126, 219, 255, 189, 165, 165, 36
50 REM Generarea culorilor si afisarea pe ecran
60 BORDER 1: PAPER 0: CLS
70 PRINT INK RND*7, "S": REM Tasta "S" accesata in
    modul de lucru G
80 GO TO 30
```

În continuare vor fi prezentate instrucţiunile FLASH, BRIGHT, INVERSE care împreună cu instrucţiunile ce asigură cromatica afişării pe ecran permit obţinerea unor efecte speciale.

FLASH

Instrucţiunea FLASH permite realizarea a ceea ce se numeşte **efect flash** şi anume afişarea cu luminozitate variabilă a informaţiei din anumite zone ale ecranului. Efectul flash constă practic în inversarea culorilor corespunzătoare cernelii şi respectiv fondului, operaţie care se repetă la intervale egale de timp.

În afară de considerente de ordin estetic, utilizarea acestei facilităţi permite ca afişarea informaţiei pe ecran să fie realizată astfel încît să se atragă atenţia asupra conţinutului anumitor zone.

FLASH poate fi indicată calculatorului fie drept comandă, fie ca instrucţiune program, fie în combinaţie cu instrucţiunile de afişaj PRINT, DRAW, CIRCLE.

Sintaxa comenzii FLASH este:

FLASH <expresie>,

iar sintaxa instrucţiunii FLASH este

<instrucţiunea FLASH> ::= {<etichetă>} FLASH <expresie>

Utilizarea facilităţii FLASH ca opţiune într-o listă corespunzătoare unei instrucţiuni de afişaj se realizează conform cu sintaxa

FLASH <expresie>.

Parametrul <expresie> este o expresie prin a cărei evaluare (cu aplicarea operaţiei de rotunjire) rezultă una din valorile 0 sau 1 sau 8.

Exemple.

```
FLASH 1
10 LET a=3: FLASH 1: PRINT a*a
10 PRINT AT 3,5; FLASH 1; "LA REVEDERE"; AT 10,5;
   FLASH 0; "PE CURIND"
```

După executarea unei comenzi sau instrucţiuni FLASH 1 toate caracterele ce vor fi înscrise pe ecran vor fi afişate în efectul flash. Executarea unei comenzi sau instrucţiuni FLASH 8 determină ca pentru poziţiile de pe ecran în care s-a specificat anterior efectul flash, respectiv poziţiile în care nu s-a dorit afişarea însoţită de acest efect să se păstreze aceste opţiuni în cazul în care alte caractere vor fi afişate în continuare în aceste poziţii.

Efectul executării unei comenzi sau instrucţiuni FLASH 0 constă în anularea efectului executării instrucţiunilor (comenzilor) FLASH 1 sau FLASH 8 precedente, după executarea acestei instrucţiuni (comenzi) toată informaţia urmînd să fie afişată în modul normal de lucru (fără efectul flash).

Utilizată ca opţiune într-o listă corespunzătoare unei instrucţiuni de afişaj, FLASH 0, FLASH 1 au efect local (ca şi în cazul utilizării opţiunilor PAPER, INK, BORDER).

Exemple:

```
10 INPUT FLASH 1: INK 0; "Cum va numiti?";a$
20 PRINT FLASH 1; AT 3,5; INK 2; PAPER 6; "ATENTIE!"
30 PRINT "abcd": FLASH 1: PRINT "ABCD":
    PRINT "1"; FLASH 0; "2"; FLASH 1; "3": PRINT "GATA"
```

•P25

```
 5 REM Trasarea mai multor cercuri rosii cu efect
   flash generat aleator, pe fond galben si chenar negru
10 BORDER 0: INK 2: PAPER 6: CLS
20 FOR x= 50 TO 200 STEP 50
30 FOR y = 50 TO 120 STEP 50
40 CIRCLE FLASH RND; x,y, 50
50 NEXT y
```

```
60 NEXT x
```

Observaţie. Dacă ultima instrucţiune de tip FLASH executată dintr-un program este FLASH 1 (sau FLASH 0), atunci după executarea programului se rămîne sub efectul dictat în instrucţiunea respectivă. De exemplu solicitarea listării programului va avea drept efect afişarea liniilor de program cu efectul flash (sau în modul normal, fără flash). Bineînţeles, ca urmare a comenzii FLASH 0 (sau FLASH 1) se poate obţine o listare a liniilor de program fără efectul flash (sau cu efectul flash).

BRIGHT

Afişarea informaţiei pe ecran poate fi realizată indicînd pentru caracterele înscrise în anumite zone ale ecranului o luminozitate mai mare (efectul brightness sau efect de "strălucire"). Acest efect poate fi realizat prin BRIGHT, care (ca şi PAPER, INK, BORDER, FLASH) poate fi utilizată, fie ca instrucţiune program, fie drept comandă (independentă de program), fie ca opţiune în combinaţie cu instrucţiunile de afişaj.

Instrucţiunea BRIGHT are structura sintactică

<instrucţiunea BRIGHT> ::= {<etichetă>} BRIGHT <expresie>

Comanda BRIGHT este formulată conform sintaxei

<comanda BRIGHT> ::= BRIGHT <expresie>

unde parametrul <expresie> este o expresie prin a cărei evaluare (cu aplicarea operaţiei de rotunjire) rezultă una din valorile 0,1 sau 8. BRIGHT utilizată ca opţiune într-o listă a unei instrucţiuni de afişaj trebuie inclusă în cadrul listei conform sintaxei

BRIGHT <expresie>

parametrul <expresie> avînd aceeaşi semnificaţie ca mai sus.

Exemple:.

```
BRIGHT 1
10 BRIGHT RND: PRINT 2+3: PRINT BRIGHT 1; "A";
   BRIGHT 0; "B"
20 PRINT BRIGHT 1; "A": PRINT BRIGHT 0; AT 3,5;
   "B"; AT 10,5; BRIGHT 1; FLASH 1; "LA REVEDERE"
```

Executarea unei instrucţiuni (comenzi) BRIGHT 1 face ca pentru toate caracterele ce vor fi în continuare afişate pe ecran să fie utilizat un grad mai mare de luminozitate (strălucire). Efectul executării unei instrucţiuni (comenzi) BRIGHT 0 constă în determinarea afişării în continuare pe ecran a informaţiei utilizînd un grad de luminozitate normal. (BRIGHT 0 anulează efectul instrucţiunilor/comenzilor BRIGHT 1 sau BRIGHT 8 precedente). Efectul executării unei instrucţiuni BRIGHT 8 este de a indica, pentru poziţiile la care au fost anterior formulate opţiuni asupra gradului de luminozitate, că în continuare aceste opţiuni trebuie să fie luate în considerare dacă pe poziţiile respective este necesară înscrierea altor caractere.

Opţiunea BRIGHT în cadrul unei liste a unei instrucţiuni de afişaj are efect local şi anume precizează gradul de luminozitate pentru afişarea informaţiei determinate de componenele din listă ce urmează opţiunii BRIGHT pînă la întîlnirea eventual a unei noi opţiuni BRIGHT.

După terminarea executării instrucţiunii de afişaj respective se revine la opţiunea BRIGHT globală (cînd nu este formulată explicit, opţiunea BRIGHT globală este BRIGHT 0).

Ca şi în cazul instrucţiunilor FLASH, PAPER, INK, BORDER, după terminarea execuţiei unui program, se păstrează în continuare efectul ultimei instrucţiuni BRIGHT executate.

Este de reţinut faptul că BRIGHT 1 determină obţinerea unei luminozităţi mai intense pentru poziţia corespunzătoare afişării unui caracter dacă toţi cei 8x8 pixeli componenţi ai celulei respective sînt coloraţi în culoarea cernelii utilizate.

●P26

```
   5 REM Generarea pe ecran a unor forme simetrice
  10 BORDER 4: PAPER 0: CLS
  20 FOR i=1 TO 4: LET p=0
  30 FOR a=1 TO 4
  40 LET x=RND*13+129
  50 FOR n=1 TO 40: GO SUB 1000: NEXT n
  60 LET i=i+1: LET p=p+1
  70 PAUSE 100
  80 NEXT a
  90 STOP
1000 LET L = INT (RND*11)
1010 LET c= INT (RND*16)
1020 INK i: PAPER p
1030 PRINT AT L,c; CHR$ x
1040 PRINT AT L, 31-c; CHR$ x
1050 PRINT AT 21-L,c; CHR$ x
1060 PRINT AT 21-L, 31-c; CHR$ x
1070 RETURN
```

Observaţie. Pentru ilustrarea modului de lucru implicat de utilizarea instrucţiunilor BRIGHT şi FLASH sugerăm introducerea liniilor de program

```
  15 BRIGHT 1
  16 FLASH 1
```

INVERSE

INVERSE permite ca afişarea informaţiei în anumite zone ale ecranului să fie realizată prin inversarea culorilor corespunzătoare cernelii şi fondului.

Deşi INVERSE poate fi utilizată şi drept comandă, cel mai frecvent îşi dovedeşte eficienţa prin considerarea ei ca instrucţiune program, respectiv ca opţiune în combinaţie cu instrucţiunile de afişaj.

Sintaxa instrucţiunii INVERSE este

<instrucţiunea INVERSE> :: =

{<etichetă>} INVERSE <expresie>

unde parametrul <expresie> este o expresie prin a cărei evaluare (cu aplicarea operaţiei de rotunjire) rezultă una din valorile 0 sau 1.

Utilizarea facilităţii INVERSE ca opţiune în lista unei instrucţiuni de afişaj se realizează conform cu sintaxa INVERSE <expresie>, unde parametrul <expresie> are aceeaşi semnificaţie ca şi în cazul instrucţiunii INVERSE.

După executarea instrucţiunii INVERSE 1, toate afişajele pe ecran determinate de executarea următoarelor instrucţiuni INPUT şi PRINT vor fi realizate utilizînd pentru cerneală culoarea selectată pentru fondul ecranului, pentru fond fiind utilizată culoarea cernelii (pînă la executarea unei instrucţiuni sau comenzi INVERSE 0). Trebuie reţinut că după executarea programului, afişarea în continuare se realizează conform cu ultima instrucţiune sau comandă INVERSE executată (inclusiv pentru afişarea liniilor programului în cazul în care după executarea programului se solicită o listare a lui). Utilizarea facilităţii INVERSE 0, sau INVERSE 1 ca opţiune într-o listă a unei instrucţiuni de afişaj are un efect local şi anume numai pentru informaţia a cărei afişare este determinată de instrucţiunea respectivă (ca şi în cazul opţiunilor PAPER, BORDER, INK, FLASH).

In cazul în care opţiunea INVERSE 1 este utilizată în combinaţie cu instrucţiunile CIRCLE, DRAW, PLOT, desenul va fi realizat în culoarea fondului, deci nu va fi vizibil pe ecran.

●P21

Reluînd un exemplu prezentat anterior, în care se include o linie de program care conţine instrucţiunea INVERSE, se obţine un desen interesant în care combinarea culorilor este generată aleator.

```
  5 REM Trasarea unor triunghiuri colorate aleator
    galben-albastru fondul ecranului fiind albastru
 10 PAPER 1: INK 6: CLS
 20 FOR y=0 TO 20 STEP 2
 40 PLOT 0,y
 30 DRAW 255, 0
 60 NEXT y
 70 FOR n=100 TO 220 STEP 30
 80 FOR x=-10-n/10 TO 10+n/10
 90 PLOT n, 35+n/10
 95 INVERSE RND
100 DRAW x, -n/4
110 NEXT x: NEXT n
```

OVER

În general, tentativa de imprimare într-o celulă a ecranului în care este înscris un caracter, a unui nou caracter, se soldează cu substituirea vechiului caracter prin cel nou indicat.

Prin includerea instrucţiunilor OVER într-un program se realizează efectul de

supraimprimare, în sensul că solicitarea de înscriere a unui caracter într-o celulă a ecranului ce conține deja un caracter conduce la suprapunerea celor două (eventual mai multe) caractere.

Instrucțiunea OVER poate fi utilizată și în modul de lucru cu rezoluție fină pentru imprimarea punctelor, segmentelor de dreaptă etc, prin considerarea culorii utilizate pentru fond în locul celei utilizate pentru cerneală.

OVER poate fi utilizată fie ca o comandă directă, fie ca instrucțiune program, fie ca opțiune într-o listă de afișaj.

Sintaxa instrucțiunii OVER este

<instrucțiunea OVER> ::= {<etichetă>} OVER <expresie>

unde parametrul <expresie> este o expresie prin a cărei evaluare (cu aplicarea operației de rotunjire) rezultă una din valorile 0 sau 1.

Efectul executării instrucțiunii OVER 1 este ca în continuare, orice solicitare de înscriere pe ecran a unui caracter într-o celulă în care este deja înscris un caracter să determine suprapunerea celor două caractere. Efectul executării instrucțiunii OVER 0 constă în anularea efectului instrucțiunii OVER 1 precedente (restabilirea modului standard de înscriere cu substituire pe ecran).

OVER poate fi utilizată ca opțiune într-o listă PRINT sau INPUT, caz în care efectul este numai local și anume indică modul de lucru (cu substituire, sau suprapunere) pentru informația al cărui afișaj este determinat de instrucțiunea respectivă. OVER ca instrucțiune sau opțiune poate fi utilizată și în modul de lucru cu rezoluție fină. Fără instrucțiunea OVER 1 (opțiunea OVER 1) în cazul în care două sau mai multe curbe se intersectează, dacă pentru trasarea curbelor nu a fost considerată aceeași culoare, atunci toate celulele care conțin cel puțin un pixel comun curbelor își modifică culoarea. Dacă a fost executată instrucțiunea OVER 1 (sau scrierea este realizată sub efectul opțiunii OVER 1) în celulele care conțin cel puțin un pixel comun la cel puțin două curbe pentru ale căror trasări au fost folosite culori diferite, va fi reprodusă culoarea fondului. In acest mod afișarea încă o dată (în exact aceleași poziții) a unor puncte, segmente etc, va determina dispariția acestora de pe ecran.

Exemplu: Prin executarea programului

```
10 PRINT AT 11,15; "TITLU"; OVER 1; AT 11,15;"-----"
```

se obține pe ecran

<u>TITLU</u>

7.3 Exemple de programe

În scopul familiarizării cititorului cu facilități de grafică și cromatică ale limbajului BASIC, prezentăm în continuare o serie de programe care sperăm că se vor dovedi utile pentru asimilarea rapidă și eficientă a diferitelor modalități de combinare a instrucțiunilor.

•P27

Programul realizează desenarea unei piramide compuse din blocuri colorate. Chenarul ecranului este afişat în diferite culori pe toată durata execuţiei. La terminarea executării programului, chenarul rămîne colorat în albastru (linia 155).

Linia 160 este ciclată indefinit pentru evitarea apariţiei mesajului "O.K" care ar strica "desenul" creat. Oprirea executării programului se face prin comanda BREAK

```
  5 REM Piramida
 10 BORDER: CLS
 20 LET b=16
 50 LET t=0
 60 LET s=0
 70 LET L=20
 80 LET t=t+b
 90 FOR n=s TO s+b*2 - 2
100 PRINT AT L,n; INK INT (RND*6) + 1; "█"
105 BORDER INT (RND*6) + 1
110 NEXT n
120 LET L=L-1
130 LET b=b-1
140 LET s=s+1
150 IF b>0 THEN GO TO 80
155 BORDER 1
160 GO TO 160
```

•P28

Programul pune în evidenţă efectele BRIGHT şi FLASH în utilizarea diferitelor culori.

```
  5 REM Cromatica cu BIRGHT si FLASH
 10 PRINT INK 4; "NORMAL ████"
 15 PRINT BRIGHT 1; INK 4; "BRIGHT ████"
 20 PRINT BRIGHTZ 1; PAPER 2; "NORMAL████ "
 25 PRINT BRIGHT 1; INK 4; PAPER 2; "BRIGHT  ████"
 30 PRINT FLASH 1; INK 4; "FLASHING████"
 35 PRINT BRIGHT 1; FLASH 1; INK 4; "BRIGHT ████"
 40 PRINT FLASH 1; PAPER 2; INK 4; "FLASHING████"
 45 PRINT FLASH 1; BRIGHT 1; PAPER 2; INK 4;
    "BRIGHT████ "
```

Programul ilustrează efectele utilizării opţiunilor FLASH şi BRIGHT în listele INPUT.

```
  5 REM Optiunile BRIGHT si FLASH în instructiunile
    INPUT
 10 INPUT PAPER 6; INK 1; "Indicati o culoare pentru
     cerneala"; i
 20 INPUT INK 2; "Indicati o culoare pentru fond"; f
 30 INPUT FLASH; BRIGHT 1; INK 4; PAPER 2; "Indicati
```

```
   un cuvint"; a$
40 LET a$ = CHR$ 16 + CHR$ i + CHR$ 17 + CHR$
   p + a$
50 PRINT AT 10, 10; a$
```

•P29•

Programul prezentat în continuare pune în evidență posibilitățile de cromatică de care dispune calculatorul pe ecran

```
  5 REM Test cromatica
 10 PAPER 7; BORDER 0: CLS
 20 LET a = RND*10
 30 LET b= RND*16
 40 LET z= RND*7
 50 PRINT AT a,b; INK z; "█"
 60 PRINT AT 21-a, b; INK z; "█"
 70 PRINT AT 21-A, 31-B; INK z; "█"
 80 PRINT AT a, 31-b; INK z; "█"
 90 IF RND>RND THEN GO TO 20
100 BEEP RND/30, RND*60 - RND*60
110 GO TO 20
```

În cadrul acestui exemplu a fost considerată instrucțiunea BEEP care determină emiterea unui sunet de o anumită înălțime și durată atunci cînd este executată linia de program 100 (vezi capitolul 8).

O variantă a acestui program care utilizează opțiunile BRIGHT, FLASH este dată de următorul exemplu. Prin compararea rezultatelor obținute la executarea celor două programe, se pot observa efectele utilizării opțiunilor prezentate.

```
  5 REM Test cromatica: optiunile BRIGHT si FLASH
 10 PAPER 7: BORDER 0: CLS
 20 LET a = RND*10
 25 LET f = RND
 30 LET b= RND*16
 40 LET z= RND*6
 50 PRINT AT a,b; FLASH f; BRIGHT 1; INK z; "█"
 60 PRINT AT 21-a,b; FLASH f; BRIGHT 1; INK z; "█"
 70 PRINT AT 21-a, 31-b; FLASH f; BRIGHT 1; INK z;"█"
 80 PRINT AT a, 31-b; FLASH f; BRIGHT 1; INK z; "█"
 90 IF RND>RND THEN GO TO 20
100 BEEP RND/30, RND*60 - RND*60
110 GO TO 20
```

Oprirea execuției se realizează prin comanda BREAK.

•P30

Programul următor pune în evidență efectele cromatice combinate cu grafica în rezuluție fină

```
 5 REM Test Galaxie
20 PAPER 0: BORDER 0: CLS
30 LET c=255: LET d=175
40 INK RND*7
50 LET a=c*RND
60 LET b=d*RND
70 PLOT a,b: PLOT a,d-b
80 PLOT c-a,b: PLOT c-a, d-b
90 IF RND>.5 THEN GO TO 60
95 INK RND*7
100 GO TO 50
```

Oprirea execuţiei programului se realizează prin BREAK.

•P31

Programul realizează trasarea unei aproximaţii a graficului funcţiei sinus utilizînd posibilităţile cromatice ale calculatorului.

```
 10 REM Grafic sinus
 30 BORDER 2: CLS
 40 FOR x=0 TO 63 STEP .5
 50 LET y=20*SIN (x/32*PI)
 60 IF y = 0 THEN GO TO 100
 70 FOR n=0 TO y STEP SGN y/4
 80 PLOT INK RND*6; x*3 + 30, 3*(n+30)
 90 NEXT n
100 BEEP.1;x : NEXT x
```

Programele următoare pun în evidenţă posibilităţi de cromatică în combinaţie cu grafica de fină rezoluţie.

•P32

Desenul rezultat prin executarea primului program este asemănător unui "geam spart".

```
 10 REM Grafica
 30 PAPER 7
 50 LET a=INT (RND*8)
 60 LET b=INT (RND*7)
 70 IF a=b THEN GO TO 60
 80 BORDER a
 90 INK b
100 CLS
110 LET c=INT (RND*256)-128
120 LET d=INT (RND*172) -85
130 PLOT 128, 86
140 DRAW c,d
145 BEEP .01 , RND*100 - 50
```

```
150 IF RND>0.02 THEN GO TO 110
160 RUN
```

•P33

Desenul rezultat prin executarea următoarelor programe este o structură "spiralată" constînd din curbe care se intersecează în acelaşi punct.

```
 10 REM Grafica Varianta 1
 30 PAPER 7
 50 LET a=INT (RND*8)
 60 LET b= INT (RND*7)
 70 IF a=b THEN GO TO 60
 80 BORDER a
 90 INK b
100 CLS
110 LET c=INT (RND*256)-128
120 LET d=INT (RND*172)- 85
130 PLOT 128,86
140 DRAW c,d,PI/2
145 BEEP . 01, RND*100 - 50
150 IF RND >0.015 THEN GO TO 110
160 RUN
```

```
 10 REM Grafica Varianta 2
 30 PAPER 7
 50 LET a= INT (RND*8)
 60 LET b=INT (RND*7)
 70 IF a=b THEN GO TO 60
 80 BORDER a
 90 INK b
100 CLS
110 LET c=INT (RND*256)-128
120 LET d=INT (RND*172)- 85
130 PLOT 128, 86
135 IF RND>0.5 THEN INK RND*6
140 DRAW c,d, PI/2
145 BEEP .01 , RND*100-50
150 IF RND >0.015 THEN GO TO 110
160 RUN
```

Oprirea execuţiei acestor programe se realizează prin BREAK.

Se observă că în cazul mai multor programe prezentate, RUN a fost utilizată ca instrucţiune program.

• P34

Cele două variante de program care urmează realizează desene tubulare asemănătoare cu un tunel.

```
 5 REM Tunel Varianta 1
15 BORDER 51: PAPER 7: CLS
20 CIRCLE INK RND*6; 128+RND*10 - RND*10,
    86 + RND*7 - RND-7, RND*65
30 IF RND>. 92THEN CLS
40 BEEP RND/3, RND*100 - 30
50 GO TO 20
```

```
 5 REM Tunel Varianta 2
20 CIRCLE INK RND*6; 128 + RND*10 - RND*10,
   86+RND*7-RND-7, RND*65
25 PLOT 128, 86
30 DRAW INK RND*6; OVER 1; 32-RND*64, 21-RND*42
40 BEEP RND/4, RND*50 - 30
50 IF RND>.33 THEN GO TO 30
60 IF RND<.93 THEN GO TO 20
70 RUN
```

●P35

Variantele de program prezentate permit obținerea unor desene "artistice" pe ecranul calculatorului.

```
   1 REM Forme generate aleator. Varianta 1.
   5 BORDER 7: PAPER 7: CLS: INK 0
  10 LET x=INT (RND*256)
  20 LET y=INT (RND*176)
  30 LET l=INT (RND*256)
  40 LET m=INT (RND*176)
  50 LET u=15: LET v=7
  60 DEF FN r(x) = INT (RND*x)
 100 GO SUB 1000
 115 LET num=num-1
 116 IF num<>0 THEN GO SUB 1000
 120 PLOT x,y
 130 DRAW l-x, m-y
 140 IF x+a>255 OR x+a<0 THEN LET a=-a
 150 IF y+b>175 OR y+b<0 THEN LET b=-b
 160 IF l+c>255 OR l+c<0 THEN LET c=-c
 170 IF m+d>175 OR m+d<0 THEN LET d=-d
 180 LET x=x+a: LET y=y+b
 190 LET l=l+c : LET m = m+d
 210 LET cont = FN r(200)
 220 IF cont = 1 THEN RUN
 230 GO TO 110
1000 LET a=FN r(u) - v
1010 LET b=FN r(u) - v
1020 LET c=FN r(u) - v
1030 LET d=FN r(u) - v
1040 LET num=FN r(20) + 10
1050 RETURN
```

```
  1 PAPER 0: INK 7: BORDER 0: CLS
  5 REM Forme generate aleator. Varianta 2
 10 FOR x=0 TO 255
 20 PLOT x,0
 30 DRAW OVER 1; 255-x*2, 175
 40 NEXT x
 50 FOR y = 0 TO 175
 60 PLOT 0,y
 70 DRAW OVER 1; 255, 175-y*2
 80 NEXT y
```

```
  1 REM Forme generator aleator. Varianta 3
  5 PAPER 0 : CLS: border 0
 10 FOR x=0 TO 255
 20 PLOT x,0
 30 DRAW OVER 1; 255-x*2, 175
 40 NEXT x
 50 FOR y=0 TO 175
 60 PLOT 0,y
 70 DRAW OVER 1; 255, 175-y*2
 80 NEXT y
 85 LET s= RND*. 5
 90 FOR x=255 TO 0 STEP-5
100 PLOT x,0
110 DRAW OVER 1; 255 -x*2, 175
120 NEXT x
130 FOR y = 0 TO 175 STEP 5
140 PLOT 0,y
150 DRAW OVER 1; 255, 175 -y*2
160 NEXT y
165 PAUSE 200
170 INK RND*7: PAPER 0: CLS
180 GO TO 85
```

8. EFECTE SPECIALE

8.1 Efecte sonore: BEEP

Majoritatea microcalculatoarelor compatibile cu calculatoarele de tip SPECTRUM, TIM-S etc. dispun de dispozitive care permit emiterea de sunete. Controlul asupra înălțimii și duratei sunetului emis este realizat prin instrucțiunea BEEP. BEEP poate fi utilizată atît ca instrucțiune program cît și în modul de comandă directă adresată calculatorului. Sunetele ce pot fi emise de către dispozitivul de sonorizare al unui microcalculator au asociate cîte un cod exprimat printr-un număr întreg din intervalul [–69,69].

Sunetele corespunzătoare la coduri exprimate prin numere consecutive diferă prin aproximativ un semiton (de exemplu, sunetelor notelor **do** diez şi **re** bemol le corespunde acelaşi cod). Codul asociat unui sunet exprimă numărul de semitonuri dintre nota **do** (din registrul normal), care are codul 0, şi nota corespunzătoare sunetului respectiv.

Durata sunetului (durata emisiei sunetului de către calculator) este exprimată printr-un număr întreg cuprins între 0 şi 10, ce exprimă numărul de secunde în care are loc emiterea sunetului.

Instrucţiunea BEEP are structura sintactică:

<instrucţiunea BEEP> :: =

{<etichetă>} BEEP <expresie 1>, <expresie 2>

unde parametrii <expresie 1>, <expresie 2> sînt expresii prin ale căror evaluări (cu aplicarea operaţiei de rotunjire) se obţin constante între 0 şi 10 pentru <expresie 1>, şi între –69 şi 69 pentru <expresie 2>.

Executarea unei instrucţiuni BEEP determină emiterea sunetului avînd codul egal cu valoarea obţinută prin evaluarea expresiei <expresie 2> pe o durată de **t** secunde, unde **t** este valoarea obţinută prin evaluarea parametrului <expresie 1>.

Exemplu:

```
10 FOR n=0 TO 100: BEEP INT (RND*10), n:NEXT n
```

•P36

```
 5 REM Emiterea gamei do major
10 BEEP 1,0: BEEP 1,2: BEEP 1,4: BEEP 1,5
20 BEEP 1,7: BEEP 1,9: BEEP 1,11;BEEP 1,12
```

•P37

```
 5 REM Fragment din marsul funebru din simfonia 1 a
   lui Mahler
20 BEEP 1,0: BEEP 1,2: BEEP. 5,3: BEEP. 5,2: BEEP 1,0
30 BEEP 1,0: BEEP 1,2: BEEP. 5,3: BEEP. 5,2; BEEP 1,0
40 BEEP 1,3: BEEP 1,5: BEEP 2,7
50 BEEP 1,3: BEEP 1,5: BEEP 2,7
60 BEEP.75,7: BEEP.25,8: BEEP 5,7: BEEP.5,5:
   BEEP.5,3: BEEP.5,2: BEEP 1,0
70 BEEP.75,7: BEEP.25,8: BEEP.5,7: BEEP.5,5: BEEP.5,3:
   BEEP.5,2: BEEP 1,0
80 BEEP 1,0: BEEP 1,-5: BEEP 2,0
90 BEEP 1,0: BEEP 1,-5: BEEP 2,0
```

•P38

```
 5 REM Genereaza aleator efecte "sunet si culoare".
10 BEEP RND/RND/3, RND*60-35
12 BORDER RND*7
15 BEEP RND/RND/2, RND*80-45
20 BORDER RND*7
```

```
25 BEEP RND/RND/3, RND*130-65
30 PAPER RND*7
40 CLS
45 BEEP RND/RND/2, RND*40-5
50 GO TO 10
```

• P39

```
 5 REM Muzica "ciclica"
10 FOR a=-60 TO 60
20 FOR b= .01 TO . 03 STEP .01
30 BEEP b,a: BEEP b, a/10*b: BEEP b, ABSa
40 NEXT b
50 NEXT a
```

•P40

```
 5 REM Muzica generata aleator
10 LET p= INT (RND*24)-12
20 LET d= (INT (RND*3)+1)/30
30 BEEP d,p
40 IF RND>=.  7 THEN GO TO 30
50 GO TO 10
```

•P41

```
 10 REM Microcalculatorul "bine temperat"
 20 DIM a(8)
 30 FOR b = 1 TO 8
 40 READ a(b)
 50 NEXT b
 70 LET b = INT(RND*8)+1
 75 LET m= (INT (RND*4)+1)/10
 80 BEEP m,a(b)
 85 IF RND>.9 THEN GO SUB 110
 90 GO TO 70
100 DATA 0,2.039, 3.86, 4.98,7.02, 8.84, 10.88,12
105 STOP
110 LET z=RND
112 LET m=a(1)*(z>=.5)+a(8)*(z <.5)
115 BEEP 1,m
120 PAUSE 25
130 RETURN
```

8.2 Animaţie: INKEY$, ATTR

Limbajul BASIC oferă facilităţi pentru obţinerea unor efecte speciale rezultate din combinarea cromaticii, a posibilităţilor de sonorizare sincronizate cu modalităţi de afişaj care să creeze *iluzia de mişcare*. In particular, este posibilă scrierea de programe pentru simularea de jocuri om – calculator sau calculator versus calculator.

În cadrul acestui paragraf vom căuta să ilustrăm aceste posibilități de utilizare combinată a facilităților de grafică, cromatică, sonorizare, în scopul creerii efectelor de tip animație.

•P42

```
  1 REM "Deplasarea" pe verticala ecranului a
    unui caracter creat de catre utilizator
  5 BORDER 3 : PAPER 5 : CLS
 10 FOR x=0 TO 7
 20 READ y
 30 POKE USR "A" + x,y: REM Tasta "A" este accesata in
     modul de lucru G
 40 NEXT x
 50 DATA 60, 126, 219, 255, 189, 165, 165, 36
 60 FOR x=0 TO 7
 70 READ y
 80 POKE USR "B" + x,y: REM Tasta "B" este accesata în
     modul de lucru G
 90 NEXT x
100 DATA 16, 16, 16, 16, 16, 16, 16, 16
110 FOR l =0 TO 20
120 PRINT AT l,3; INK 0; "B": REM Tasta "B" accesata în
     modul de lucru G
130 PRINT AT l+1,3; INK 2; "A": REM Tasta "A" accesata
     în modul de lucru G
140 NEXT l
```

Programul realizează crearea a două noi caractere utilizînd tastele corespunzătoare literelor "A", respectiv "B" (accesate bineînțeles în modul de lucru G). Caracterele nou create sînt asemănătoare unui "păianjen", cel de al doilea caracter este utilizat pentru reprezentarea "firului" pe care se deplasează "păianjenul". Efectul de animație este creat de bucla FOR–NEXT din liniile de programe 110 și 140 care realizează simularea "deplasării" păianjenului pe ecran.

În cazul multor jocuri pe calculator este necesară simularea unei acțiuni de tip "explozie" sau generarea unui anumit efect sonor în situația în care două forme se ating (de exemplu, o "rază" care lovește un "obstacol"). Detectarea coliziunilor dintre două forme pentru care este generat cîte un tip de mișcare, este relativ simplu de realizat și anume prin includerea în program a unor testări asupra coincidenței coordonatelor pozițiilor punctelor (sau caracterelor) aflate în "mișcare" pe ecran (atît în modul de lucru de mică putere rezolutivă cît și în cazul modului de lucru cu rezoluție fină).

Următorul exemplu simulează deplasarea simultană a două obiecte pe ecran, tipul de mișcare pentru fiecare dintre ele fiind mersul la întîmplare în celulele unei rețele bidimensionale în care sînt generate celule de tip "obstacol". Convențional vom numi cele două obiecte "șoarece" și respectiv "pisică" și le vom reprezenta prin caracterele "S", respectiv "P".

Rețeaua bidimensională (labirintul) va corespunde celulelor ecranului (în modul

de lucru de mică rezoluţie) obstacolele fiind reprezentate prin celule care conţin caracterul "X". La fiecare moment, fiecare dintre cele două obiecte alege la întîmplare una din ieşirile posibile (inclusiv posibilitate de a rămîne în continuare în aceeaşi celulă). Jocul se termină fie cînd este indicată comanda BREAK, fie cînd şoarecele şi pisica se află simultan în aceeaşi celulă a ecranului.

•P43

```
  5 REM Soarecele si pisica
  6 BORDER 0 : LET b=INT(RND*7):PAPER b:INK7-b
  7 REM Coeficientul r determina proportia obstacolelor
     in retea
  8 INPUT "Dati coeficientului r, 0<r<1 pentru
    determinarea tipului de labirint" ;r
 10 INPUT "Pozitia initiala a pisicii linie ="; lp,
    "coloana="; cp
 20 IF ((lp<0)+(lp>21)+(cp<0)+(cp>31)<>0 THEN PRINT
     "Ati gresit,repetati va rog": GO TO 10
 30 INPUT "Pozitia initiala a soarecelui linie =" ; ls
    "coloana ="; cs
 35 IF ((ls<0)+(ls>21)+(cs<0)+(cs>31))<> 0 THEN PRINT
    "Ati gresit,repetati va rog": GO TO 30
 36 CLS
 40 FOR i=0 TO 21 : FOR j=0 TO 31: IF RND< r THEN
    PRINT AT i,j; "X"
 45 NEXT j: NEXT i
 50 PRINT AT lp, cp; "P"; ls, cs; "S"
 55 REM Generarea deplasarilor pentru pisica si soarece
 56 DIM p (2,9): DIM s(2,9)
 60 LET kp = 0 : LET ks = 0
 65 FOR x = -1 TO 1 : FOR y = -1 TO 1
 70 IF (ls+x<0)+(ls+x>21) <> 0 THEN GO TO 100
 80 IF (cs+y<0)+(cs+y>31) <> 0 THEN GO TO 95
 90 IF SCREEN$ (ls+x, cs+y)<>"X"THEN LET ks=ks+1:
     LET s(1,ks)= ls+x: LET s (2,ks)=cs+y
 95 NEXT y
100 NEXT x
105 FOR x =-1 TO 1 : FOR y =-1 TO 1
110 IF (lp+x<o)+(lp+x>21)<>0 THEN GO TO 130
115 IF (cp+x<0)+(cp+y>31)<>0 THEN GO TO 125
120 IF SCREEN$ (lp+x, cp+y)<> "X" THEN LET kp=kp+1:
     LET p (1,kp) = lp+x: LET p (2,kp) = cp+y
125 NEXT y
130 NEXT x
135 IF kp=1 THEN LET nlp=lp: LET ncp=cp: GO TO 145
140 LET a=INT (RND*kp)+1: LET nlp=p (1,a): LET ncp=p (2,a)
145 IF ks=1 THEN LET nls=ls: LET ncs = cs: GO TO 155
150 LET a=INT(RND*ks)+1: LET nls=s (1,a): LET ncs=s (2,a)
155 PRINT AT ls, cs; "@"; AT lp, cp;"@"; AT nls, ncs;
     "S"; AT nlp, ncp; "P"
160 IF (nls=nlp)+(ncs=ncp)=2 THEN CLS: PRINT FLASH 1;
```

```
    "PISICA A PRINS SOARECELE": PRINT "pozitia linie=";
    nls, "coloana="; ncs: BEEP RND*3, RND*69: GO TO
    170
165 LET ls=nls: LET cs=ncs: LET lp=nlp: LET
    cp=ncp: BEEP RND*10, RND*(-68): GO TO 60
170 INPUT "Doriti reluarea?(da/nu)"; a$
180 IF a$ <> "da" THEN GO TO 250
200 INPUT "Doriti aceeasi retea?(da\nu)";b$
210 IF b$="nu" THEN BEEP 2,10: GO TO 6
220 INPUT "Dati noua pozitie a pisicii linie =";
    lp, "coloana="; cp
225 IF ((lp<0)+(lp>21)+(cp<0)+(cp>31) <>0 THEN PRINT
    "Ati gresit, repetati va rog": GO TO 220
230 INPUT "Dati noua pozitie pentru soarece linie=";
    ls, "coloana="; cs
235 IF ((ls<0)+(ls>21)+cs<0)+(cs>31) <>0 THEN PRINT
    "Ati gresit, repetati va rog": GO TO 230
240 LET b = INT (RND*7)+1: PAPER b: INK 7-b: BEEP 2,
    -10: GO TO 50
250 STOP
```

INKEY$

Pentru programarea problemelor de tip joc, limbajul dispune de o funcție specială deosebit de utilă: funcția INKEY$. Funcția INKEY$ nu necesită argumente și permite detectarea intervențiilor utilizatorului prin acționarea diferitelor taste ale claviaturii terminalului în timpul desfășurării execuției programului. In general, cu excepția comenzilor BREAK, CONTINUE, intervenția utilizatorului prin acționarea diferitelor taste nu afectează desfășurarea execuției programului. Executarea unei instrucțiuni INPUT determină calculatorul să "aștepte" pînă cînd utilizatorul introduce datele prin intermediul tastaturii, pentru fiecare variabilă din lista INPUT cîte o expresie. După ce datêle au fost introduse este încheiată executarea instrucțiunii INPUT și se continuă executarea următoarelor instrucțiuni din program.

Funcția INKEY$ creează posibilitatea ca anumite date să fie introduse prin tastatură în timpul executării diferitelor instrucțiuni, calculatorul avînd posibilitatea de a reține caracterul corespunzător ultimei taste acționate. Apelarea funcției INKEY$ furnizează ca rezultat acest caracter. Dacă pe parcursul execuției programului nu a fost acționată de către utilizator nici o tastă, atunci apelarea funcției INKEY$ furnizează ca rezultat stringul vid. Trebuie reținut faptul că INKEY$ face distincția între caractere de tip literă format mic și majuscule. Funcția INKEY$ este în general utilizată pentru verificarea apariției unei anumite opțiuni din partea utilizatorului permițînd în același timp și atribuirea unui caracter ca valoare unei variabile de tip string. Acest lucru se poate realiza, de exemplu, astfel:

```
70 LET a$ = INKEY$
80 IF INKEY$ = "N" THEN STOP
90 IF INKEY$ <> "y" THEN GO TO 60
```

Exemplu.

```
 5 REM Test verificare mod de lucru cu functia INKEY$
10 LET a$ = ""
20 FOR i=1 TO 100 : LET a$=a$ + INKEY$: PAUSE 0:
   PRINT INKEY$ : NEXT i
30 PRINT "a$= " ; a$
```

Programul afişează caracterele corespunzătoare tastelor acţionate de către utilizator în timpul execuţiei programului.

Deoarece execuţia programului se realizează extrem de rapid în comparaţie cu viteza cu care utilizatorul poate acţiona diferitele taste, a fost utilizată instrucţiunea PAUSE 0 pentru ca executarea programului să se desfăşoare în ritmul dorit de utilizator.

Reamintim că PAUSE 0 determină suspendarea executării instrucţiunilor din program pe o durată nedefinită, momentul reluării execuţiei fiind determinat de către utilizator prin acţionarea unei taste oarecare. Dacă în locul instrucţiunii PAUSE 0 este considerată instrucţiunea PAUSE 150 efectul executării programului este acelaşi dacă acţionarea tastelor de către utilizator este suficient de rapidă. Explicaţia constă în aceea că PAUSE 150 determină întreruperea continuării executării instrucţiunilor din program pe o durată de 3 secunde, intervenţia utilizatorului manifestată prin acţionarea unei taste anihilînd efectul acestei instrucţiuni şi determinînd astfel reluarea executării instrucţiunilor din program eventual după mai puţin de 3 secunde.

•P44

```
 5 REM Programul determina deplasarea caracterului "*"
   pe ecran prin actionarea tastelor "1", "2", "3", "4"
 6 REM Tasta "1" determina deplasarea cu o pozitie
   la stinga; tasta "2" determina deplasarea cu
   o pozitie la dreapta
 7 REM Tastei "3" determina deplasarea cu o pozitie
   in sus; actionarea tastei "4" determina deplasarea
   cu o pozitie în jos
10 INPUT "Indicati pozitia initiala linie =";
   l, "coloana="; c
15 IF (l<0)+(l>21)+(c<o)+(c>31) <> 0 THEN PRINT
   AT 10, 10; "Ati gresit, repetati": GO TO 10
16 CLS
17 PRINT AT l,c; "*"
20 PAUSE 0
25 IF INKEY$ = "1" THEN IF l<>0 THEN LET l=l-1
30 IF INKEY$ = "2" THEN IF l<>21 THEN LET l=l+1
35 IF INKEY$ = "3" THEN IF c<>0 THEN LET c=c-1
40 IF INKEY$ = "4" THEN IF c<>31 THEN LET c=c+1
50 GO TO 17
```

• P45

```
 5 REM Detectarea caracterului corespunzator tastei
   actionate
```

```
10 PAUSE 0
20 LET a$ = INKEY$
30 PRINT "Ati actionat caracterul"; a$
40 IF a$ = "0" THEN STOP
50 GO TO 10
```

Următoarele exemple ilustrează modalităţi de utilizare combinată a instrucţiunilor de grafică, cromatică şi sonorizare.

•P46

Programul prezentat în continuare simulează jocul cunoscut sub numele de "Singuraticul".

La începutul jocului sînt vizualizate pe ecran 33 de poziţii dintre care 32 conţin acelaşi caracter reprezentînd o "piesă", poziţia din centrul ecranului fiind liberă. Efectuarea unei mutări constă în deplasarea uneia dintre piese (vertical sau orizontal) peste una din poziţiile ce conţin o piesă a jocului astfel încît să se ajungă într-un spaţiu liber. Piesa peste care a fost realizată deplasarea este eliminată de pe ecran. Jucătorul cîştigă jocul dacă reuşeşte să indice o secvenţă de deplasări, astfel încît să se ajungă la configuraţia în care rămîne o singură piesă plasată pe poziţia din centru. Programul indică la fiecare etapă a jocului numărul mutării şi numărul pieselor rămase. Jucătorul indică mutarea pe care o doreşte indicînd coordonatele piesei pe care intenţionează să o deplaseze, printr-un număr de 2 cifre, comanda ENTER şi apoi tastarea numărului de două cifre reprezentînd coordonatele poziţiei în care trebuie deplasată piesa respectivă (urmată de comanda ENTER). La momentul iniţial tabla pe care se desfăşoară jocul este reprezentată prin configuraţia

1	2	3	4	5	6	7	
		X	X	X			–1
		X	X	X			–2
X	X	X	X	X	X	X	–3
X	X	X		X	X	X	–4
X	X	X	X	X	X	X	–5
		X	X	X			–6
		X	X	X			–7

Pentru a indica deplasarea piesei din celula de pe coloana 6 şi linia 4 în spaţiul din centrul tablei de joc, va trebui indicată secvenţa de comenzi 64 ENTER 44 ENTER.

```
10 REM Singuraticul
20 GO SUB 260
30 GO SUB 180
50 INPUT "Indicati piesa pe care doriti sa
   o deplasati"; a
60 BEEP.1, RND*20+20: IF a(a)<>144 THEN GO TO 50
```

```
 70 INPUT TAB 5; (a);"unde doriti sa o deplasati?"; b
 80 BEEP.1,20 + RND*20: IF a(b)<>e THEN GO TO 70
 90 LET a ((a+b)/2)=e: LET a(a)=e: LET a(b) = 144
100 LET m=m+1
110 LET c=0
120 FOR f=11 TO 75
130 IF a(f)=144 THEN LET c=c+1
140 NEXT f
150 GO SUB 180
160 IF c=1 AND a(44)= 144 THEN PRINT FLASH 1;
    TAB 5; "Ati efectuat "; m; " mutari!": STOP
170 PRINT AT 18,0; "Pe tabla de joc se afla ";
   INVERSE 1; "¤"; c; "¤"; INVERSE 0;"piese":
   GO TO 40
190 PRINT AT 7,6; INK 4; "1234567"; AT 14,13; " -7";
    AT 8,6;
200 FOR d=11 TO 75
210 IF d - 10*(INT (D/10))=8 THEN LET d=d+2: PRINT INK
    4; " - "; INT (d/10) -1; TAB 6; : GO TO 230
215 IF a(d)=144 THEN BEEP. 005,50
216 IF a(d)=e THEN BEEP.04, RND*5
220 PRINT INK (RND*5+1)*(a(d) = 144); CHR$ (a(d));
230 NEXT d
240 PRINT AT 3,2; PAPER RND*7; "¤¤"; m; " pina in acest
    moment"
250 RETURN
260 BORDER 2: PAPER 7: BRIGHT 1: INK 0: CLS: DIM a(87)
270 LET b= 32: LET a = 144: LET e = 143
280 FOR d=11 TO 75: IF d-10*(INT(d/10))=8 THEN LET
    d=d+3
290 READ a(d)
300 NEXT d
310 LET m=0
320 FOR z = 0 TO 6
330 READ x
340 POKE USR "A" + z, x
345 NEXT z
350 RETURN
360 DATA b,b,a,a,a,b,b
370 DATA b,b,a,a,a,b,b
380 DATA a,a,a,a,a,a,a
390 DATA a,a,a,e,a,a,a
400 DATA a,a,a,a,a,a,a
410 DATA b,b,a,a,a,b,b
420 DATA b,b,a,a,a
430 DATA 28,93,73,62,28,54,99,0
```

•P47

Programul următor este un joc-test pentru verificarea rapidității cu care utilizatorul poate acționa asupra tastelor claviaturii. După lansarea în execuție a programului va apare pe ecran afișat "PREGATIȚI-VA, INCEPEM" după care va apare mesajul "GATA, APASATI CIT MAI RAPID TASTA Z" moment de la care începînd calculatorul începe să numere de cîte ori acționați tasta respectivă. Calculatorul realizează compararea performanței pe care o realizați cu cea mai bună performanță pe care ați avut-o repetînd acest joc, indicîndu-vă, de asemenea, și un calificativ pentru rapiditatea de care ați dat dovadă. Jocul continuă pînă cînd obțineți un calificativ inferior valorii 5 (sau pînă cînd suspendați executarea programului prin comanda BREAK).

```
  5 REM JOC-TEST
 10 LET hs = 1000
 15 RANDOMIZE: DIM b$ (32)
 20 PRINT AT 5,5; "PREGATITI-VA, INCEPEM"
 30 FOR a = 1 TO 200 + RND * 500
 40 NEXT a
 50 PRINT AT 5,5; FLASH 1; INK 2; PAPER 6; "GATA,
    APASATI CIT MAI RAPID TASTA Z "
 60 LET c=0
 70 LET c= c+1
 90 LET a$ = INKEY$
100 IF a$ <> "Z" THEN GO TO 70
110 PRINT "SCORUL A FOST "; c
120 IF c<hs THEN LET hs = c: BEEP 1, c*(c<60)
130 PRINT AT 0,0; b$; INK 1; INVERSE 1; "CEL MAI BUN
    SCOR PINA IN PREZENT A FOST "; hs; AT 0,0; INK 2;
    b$ (TO hs * (hs<32))
135 IF hs<5 THEN GO TO 160
140 IF INKEY$ <> ""THEN GO TO 140
150 GO TO 20
160 FLASH 1: INK 2: PAPER 6: CLS
170 PRINT AT 10,10; "SINTETI CAMPION!"
```

•P48

Programul ce va fi prezentat ar putea fi numit "Colecționarul". Jocul constă în colecționarea obiectelor expuse pe ecran sub forma unor pete colorate.

Utilizatorul trebuie să dirijeze "colecționarul" astfel încît acesta să reușească să adune cît mai multe dintre obiectele expuse pe ecran. Dirijarea "colecționarului" este realizată prin utilizarea tastelor 5,6,7,8 corespunzînd direcției dorite. Un obiect este considerat ca fiind "colecționat" dacă "colecționarul" atinge celula corespunzătoare obiectului respectiv. Timpul în care jocul se desfășoară este limitat, la terminarea jocului fiind afișată pe ecran performanța realizată .

```
 10 REM COLECTIONARUL
 20 GO SUB 1000
 30 REM In linia 50 tasta M va fi în modul de lucru G
 40 FOR q = 1 TO 50
```

```
  50 PRINT AT RND * 16+3, RND*26+4; INK 1; "M"
  60 NEXT g
  70 LET x=0
  80 LET y=0
  85 FOR g=1 TO 500
  90 LET a=x
 100 LET b=y
 110 LET y=y - (INKEY$="7" AND y>1)+(INKEY$="6" AND
     y<20)
 120 LET x=x - (INKEY$ = "5" AND x>1) + (INKEY$="8" AND
     x<31)
 130 IF ATTR (y,x)=49 THEN LET s=s+1: PRINT AT 0,20;
     INK RND*7; FLASH 1; "SCORUL ESTE "; s; AT 20,0;
     "TIMP DISPONIBIL ="; 500 - g
 140 IF ATTR (y,x)=49 THEN BEEP .01, 40
 150 REM Tasta D trebuie actionata in modul de lucru
     G în linia de program 160
 160 PRINT AT b,a; "¤¤"; AT y,x; INK 2; "D"
 170 NEXT g
 180 PRINT AT 0,20; INK RND*7; FLASH 1; "SCORUL ESTE ";
     s; AT 20,0; "TIMP DISPONIBIL="; 0
 190 BEEP.1, RND*50 : GO TO 190
 900 STOP
1000 FOR a=0 TO 7
1010 READ q
1020 POKE USR "M" + a,q
1030 NEXT a
1040 DATA BIN 10100101, BIN 01011010, BIN 10100101,
     BIN 01011010, BIN 01011010, BIN 10100101,
     BIN 01011010, BIN 10100101
1050 FOR a=0 TO 7
1060 READ q
1070 POKE USR "D" + a, q
1080 NEXT a
1090 DATA BIN 00011111, BIN 01111111, BIN 11111100,
     BIN 11111000, BIN 11100000, BIN 11111000,
     BIN 01111100, BIN 00111111
1100 BORDER 2 : PAPER 6 : CLS
1120 LET s=0
1500 RETURN
```

Observație. În linia de program 140 a fost utilizată funcția ATTR care permite obținerea atributelor unei celule specificate a ecranului.

Atributele unei celule a ecranului sînt: culorile corespunzătoare cernelii și fondului, gradul de luminozitate și, dacă pentru afișarea unui caracter în celula respectivă este indicat, efectul flash.

ATTR

Sintaxa apelării funcţiei ATTR este :

<funcţia ATTR>::= ATTR (<expresie 1>, <expresie 2>)

unde parametrii <expresie 1>, <expresie 2> sînt expresii prin ale căror evaluări se obţin date de tip numeric. Evaluările celor două expresii se realizează cu aplicarea operaţiei de rotunjire, pentru <expresie 1> trebuie să rezulte o valoare între 0 şi 23, iar pentru <expresie 2> o valoare între 0 şi 31.

Rezultatul apelării funcţiei ATTR este un număr întreg cuprins între 0 şi 255 care se calculează după următoarea regulă :

Fie **v,h** valorile obţinute prin evaluarea parametrilor <expresie 1>, respectiv <expresie 2>, **c** codul corespunzător cernelii, **p** codul corespunzător fondului. Avem

b = 0 dacă pentru celula indicată este specificată afişarea cu luminozitate normală (BRIGHT 0)

b = 64 dacă pentru celula indicată este specificată afişarea cu luminozitate crescută (BRIGHT 1)

f = 0 dacă pentru afişarea informaţiei în celula respectivă nu este indicat efectul flash

f = 128 altfel

Efectul apelării funcţiei ATTR va fi numărul c+8p+b+f

Exemplu: Dacă pentru celula de coordonate 11, 16 atributele sînt

c=3, p=6, b=64, f=0, atunci ATTR (11,16) este 115.

8.3 Afisaj: rotaţii, scalări, scroll

În cadrul acestui paragraf vor fi prezentate cîteva modalităţi care permit obţinerea unor efecte de deplasare a informaţiei afişate pe ecran (efect **scroll**).

Exemplul 1: Subprogramul realizează ştergerea informaţiei afişate pe un număr oarecare de linii ale ecranului permiţînd astfel obţinerea de spaţiu liber în porţiunea superioară a ecranului, care poate fi eventual utilizată pentru afişarea unor mesaje, cum ar fi scorul în desfăşurarea unui joc, instrucţiuni pentru jucător etc. Subprogramul poate fi apelat prin instrucţiunea GO SUB 8010.

```
8010 INPUT "Indicati cite linii doriti sa
     stergeti"; c
8020 IF c<0 OR c>21 THEN GO TO 8010
8030 FOR f = 21 TO 21-c STEP-1
8040 PRINT AT f,0; "¤"
8050 NEXT f
8060 PRINT AT f+1, 0;
8070 RETURN
```

Exemplul 2 : Programele scrise în cod maşină pentru realizarea operaţiei de copiere în bloc a conţinutului unei zone de memorie în altă zonă de memorie sînt mult mai performante comparativ cu programele BASIC care realizează aceeaşi operaţie. O modalitate de obţinere a efectului de deplasare (scroll) pe ecran, utilizînd limbajul BASIC, constă în memorarea imaginii sub forma unei secvenţe de caractere şi de aplicare a operaţiei dorite secvenţei astfel rezultate. Următoarele programe determină efectul scroll respectiv în direcţiile sus, jos, stînga, dreapta.

●P49

```
 5 REM Program pentru realizarea efectului scroll
   in sus
10 DIM a$ (704)
20 INPUT a$
30 PRINT AT 0,0; a$
40 LET a$ = a$ (33 TO) + "¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤
   ¤¤¤¤¤"
50 GO TO 30
```

```
 5 REM Program pentru realizarea efectului scroll
   in jos
10 DIM a$ (704)
20 INPUT a$
30 PRINT AT 0,0 ;a$
40 LET a$ = "¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤" + a$
   (TO 672)
50 GO TO 30
```

```
 5 REM Program pentru realizarea efectului scroll
   catre stinga
10 DIM a$ (704)
20 INPUT a$
30 PRINT AT 0,0; a$
40 FOR f=1 TO 673 STEP 32
50 LET a$ (f TO f+31) = a$ (f+1 TO f+31) + "¤"
60 NEXT f
70 GO TO 30
```

```
 5 REM Program pentru realizarea efectului scroll
   catre dreapta
10 DIM a$ (704)
20 INPUT a$
30 PRINT AT 0,0; a$
40 FOR f=1 TO 673 STEP 32
50 LET a$ (f TO f+31) = "¤" + a$ (f TO f+30)
60 NEXT f
70 GO TO 30
```

Exemplul 3: Programele următoare realizează un efect de tip "rotaţie" pe ecran în sensul că o linie (coloană) ce dispare la una din frontierele ecranului va fi

realizată la frontiera opusă.

•P50

```
 5 REM Program efect "rotatie" în sus
10 DIM a$ (704)
20 INPUT a$
30 PRINT AT 0,0 ; a$
40 LET a$ = a$ (33 TO) + a$ (TO 32)
50 GO TO 30
```

```
 5 REM Program efect "rotatie" în jos
10 DIM a$ (704)
20 INPUT a$
30 PRINT AT 0,0; a$
40 LET a$ = a$ (673 TO) + a$ (TO 672)
50 GO TO 30
```

```
 5 REM Program efect "rotatie" catre stinga
10 DIM a$ (704)
20 INPUT a$
30 PRINT AT 0,0; a$
40 FOR f=1 TO 673 STEP 32
50 LET a$(f TO f+31)= a$(f+1 TO f+31) + a$(f)
60 NEXT f
70 GO TO 30
```

Pentru obținerea unui efect de tip "rotație" la dreapta în programul precedent linia de program 50 trebuie înlocuită cu

```
50 LET a$ (f TO f+31)= a$ (f+31) + a$ (f TO f+30)
```

Exemplul 4: Prin utilizarea instrucțiunii POKE 23629, –1 se pot obține efecte de tip scroll care se realizează în mod mult mai rapid. Pentru ilustrare, prezentăm un program care realizează afișarea la "scară mare" a unei secvențe de cel mult 4 caractere ce pot fi eventual create de către utilizator sau caractere din setul de caractere BASIC. De asemenea, programul realizează efectul scroll în direcția către partea superioară a ecranului.

•P51

```
  1 REM SCROLL RAPID
  2 FOR m=1 TO 12: PRINT: NEXT m: POKE 23629,-1
 10 BORDER 0
 90 INPUT a$: REM Secventa a$ contine cel mult 4
    caractere
 91 FOR v = 1 TO LEN a$
 92 LET b$ = a$ (v)
100 PRINT AT 10,10; b$
110 FOR m=87 TO 80 STEP -1
120 FOR n=95 TO 88 STEP -1
130 IF POINT (m,n) = 1 THEN BEEP 0.005, n+m-150: PRINT
     INK v; AT 108-n, m + (v-1)*8 - 80 ; "█"
140 NEXT n
150 NEXT m
```

```
160 NEXT v
170 PRINT AT 10,10, "¤"
180 FOR m=0 TO 31: PRINT INK 5; AT 21,m; "█": NEXT m
190 FOR m=21 to 12 STEP-1:PRINT INK 5;AT m,31; "█":
    NEXT m
200 FOR m=31 TO 0 STEP-1:PRINT INK 5;AT 12,m; "█":NEXT m
210 FOR m=12 TO 21: PRINT INK 5; AT m,0; "█": NEXT m
215 POKE 23629, -1
220 FOR m=1 TO 21 : PRINT: NEXT m
230 GO TO 10
```

Observaţie. Efectele de sunet şi culoare în afişarea literelor la scară mare pot fi eventual "randomizate" înlocuind linia 130 cu

```
    130 IF POINT (m,n) = 1 THEN BEEP 0.005, INT (RND*k):
        PRINT INK INT (RND*5)+1
```

unde $-68 \leq k \leq 69$.

De asemenea, durata 0.005 în instrucţiunea BEEP poate fi înlocuită printr-un număr mai mare, ceea ce însă ar conduce la obţinerea unui program cu execuţie lentă. Programul poate fi completat cu subprograme în care sînt create de către utilizator alte noi caractere, instrucţiunea INPUT din linia de program 90, fiind eventual înlocuită printr-o instrucţiune de atribuire care să determine înregistrarea în variabila a$ a unei secvenţe de cel mult 4 caractere printre care pot figura inclusiv caracterele nou create.

O variantă de program care să ofere efecte cromatice interesante poate fi obţinută incluzînd linia de program

```
    15 PAPER 0
```

şi înlocuind linia 130 cu

```
   130 IF POINT (m,n)=0 THEN BEEP 0.005, n+m-150:
       PRINT INK v ; AT 108-n, m+(v-1)*8 - 80; "█"
```

II. APLICAŢII ÎN BASIC

9. Elemente de calcul matriceal

În cadrul multor aplicaţii, apare frecvent problema prelucrării datelor structurate sub forma de **matrice**. În cadrul acestui paragraf, vor fi prezentate cîteva dintre operaţiile elementare efectuate asupra matricelor. Fiecare dintre aceste operaţii va fi prezentată ca un subprogram BASIC, astfel încît o serie de aplicaţii să devină realizabile printr-o simplă asamblare a subprogramelor respective în cadrul unui program.

9.1 Introducere date

Citirea dimensiunilor unei matrice si a componentelor sale

•P52

```
3010 REM ******************************************
3015 REM * Subprogram  pentru citirea dimensiunilor  *
3020 REM  * NL,NC a  unei matrice A si a componentelor     *
3021 REM  *  sale                                         *
3025 REM ******************************************
3030 INPUT " Numarul liniilor = ";nl
3040 INPUT " Numarul coloanelor = ";nc
3050 DIM  a(nl,nc)
3060 PRINT " Componentele matricei linie cu linie "
3070 FOR i=1 TO nl
3075 PRINT " Componentele liniei  ";i
3080 FOR j=1 TO nc
3090 INPUT (" Coloana ";j);a(i,j)
3095 NEXT j
3097 NEXT i
3099 RETURN
```

9.2 Transpusa unei matrice

•P53

```
3120 REM ******************************************
3125 REM * Subprogram  pentru calculul matricei  T,   *
3126 REM * transpusa matricei A.                      *
3127 REM * A are NL linii si NC coloane               *
3130 REM ******************************************
3135 DIM t(nc,nl)
```

```
3140 FOR i=1 TO nl
3145 FOR j=1 TO nc
3150 LET t(j,i)=a(i,j)
3155 NEXT j
3160 NEXT i
3170 RETURN
```

9.3 Suma a două matrice

●P54

```
3210 REM ********************************************
3215 REM * Subprogram  pentru calculul matricei C,    *
3216 REM * suma matricelor A si B                     *
3217 REM * A are NL linii, NC coloane                 *
3218 REM * B are N1L linii,N1C coloane     .          *
3219 REM ********************************************
3220 IF (nl<>n1l) OR (nc<>n1c) THEN GO TO  3295
3230 DIM c(nl,nc)
3240 FOR i=1 TO nl
3250 FOR j=1 to nc
3260 LET c(i,j)=a(i,j)+b(i,j)
3265 NEXT j
3270 NEXT i
3280 RETURN
3285 REM *********************************************
3286 REM *Tratarea erorii datorate incompatibilitatii *
3287 REM *  dimensiunilor celor doua matrice          *
3290 REM *********************************************
3295 PRINT " Matricele nu sint compatibile pentru
      aceasta operatie "
3299 RETURN
```

9.4 Produsul a două matrice

●P55

```
3310 REM *********************************************
3311 REM * Subprogram  pentru calculul matricei C=A*B *
3312 REM * A matrice cu NL1 linii si NC1 coloane      *
3313 REM * B matrice  cu NL2 linii si NC2 coloane     *
3314 REM *********************************************
3320 IF nc1<>nl2 THEN GO TO 3380
3325 DIM c(nl1,nc2)
3330 FOR i=1 TO nl1
3335 FOR j=1 TO nc2
3340 LET c(i,j)=0
3345 FOR k=1 TO nc1
```

```
3350 LET c(i,j) =c(i,j)+a(i,k)*b(k,j)
3355 NEXT k
3360 NEXT j
3365 NEXT i
3370 RETURN
3375 REM ******************************************
3376 REM* Tratarea erorii datorate incompatibilitatii *
3377 REM *dimensiunilor matricelor A,B               *
3378 REM ******************************************
3380 PRINT " Operatia produs nu este definita pentru
     matricele considerate "
3390 RETURN
```

9.5 Puterea M a unei matrice

●P56

```
3400 REM ******************************************
3401 REM * Subprogram  pentru calculul puterii M a    *
3402 REM * matricei A cu N linii si N coloane.        *
3403 REM * Rezultatul este matricea C.                *
3404 REM ******************************************
3410 INPUT " Valoarea lui M =";m
3415 IF m<=0 THEN GO TO 3550
3420 DIM b(n,n)
3425 DIM c(n,n)
3430 FOR i=1 TO n
3435 FOR j=1 TO n
3440 LET b(i,j)=a(i,j): LET c(i,j)=a(i,j)
3445 NEXT j
3450 NEXT i
3455 IF m=1 THEN GO TO 3500
3460 LET nl1=n:LET nc1=n:LET nl2=n :LET nc2=n
3470 FOR k=1 to m-1
3475 GOSUB 3310
3480 FOR i=1 TO n
3485 FOR j=1 TO n
3487 LET b(i,j)=c(i,j)
3488 NEXT j
3489 NEXT i
3490 NEXT k
3500 REM ******************************************
3501 REM * Afisarea rezultatului                      *
3502 REM ******************************************
3505 PRINT "Matricea rezultat "
3506 PRINT "****************":PRINT:PRINT
3510 FOR i=1 TO n
3515 FOR j=1 TO n
3520 PRINT c(i,j);"¤";
```

```
3525 NEXT j
3526 PRINT
3530 NEXT i
3535 RETURN
3550 REM ****************************************
3551 REM * Datele de intrare sint incorecte     *
3552 REM ****************************************
3555 PRINT "Valoarea  m = ";m;"  este incorecta "
3560 INPUT "Doriti reluarea ?(d/n)";a$
3565 IF a$="d" THEN GOTO 3410
3570 RETURN
```

9.6 Determinantul unei matrice

•P57

```
  1 REM CALCULUL DETERMINANTULUI UNEI MATRICE PATRATE
  5 INPUT "INDICATI DIMENSIUNEA MATRICEI ";N
  6 DIM A (N,N)
 10 PRINT "INTRODUCETI LINIILE MATRICEI";AT 10,0;"APASATI ORICE
    TASTA CIND DORITI SA INCEPETI INTRODUCEREA DATELOR"
    : PAUSE 0: CLS
 15 FOR I=1 TO N: PRINT "LINIA ";I: FOR J=1 TO N:INPUT ("A(";
    I; ",";J; ")= ");A(I,J): NEXT J: NEXT I
 20 CLS: PRINT AT 1,3; INVERSE 1; "MATRICEA COEFICIEN-
    TILOR": PRINT : PRINT
 25 FOR I=1 TO N : FOR J=1 TO N: PRINT A(I,J);" ";:
    NEXT J: PRINT : PRINT I
 30 PRINT AT 20,1; FLASH 1;"APASATI ORICE TASTA DACA
    DORITI CONTINUAREA": PAUSE 0
 40 FOR I=1 TO N-1
 50 IF A (I,I)<> 0 THEN GO TO 100
 55 FOR K=I+1 TO N:IF A(K,I)<>0 THEN GO TO 70
 60 NEXT K
 62 PRINT AT 10,10; FLASH 1;"DETERMINANTUL ESTE EGAL
    CU 0"
 64 INPUT "DORITI RELUAREA PROGRAMULUI ?(DA/NU)";A$:
   IF A$="DA" THEN CLS : GO TO 5
 65 CLS: PRINT AT 10,10; FLASH 1;"LA REVEDERE !": FOR
    i=1 TO 5 BEEP RND*10,30*i-90: NEXT i: CLS : STOP
 70 FOR J=I TO N: LET D=A(I,J): LET A(I,J)=A(K,J): LET
    A(K,J)=D: NEXT J
100 FOR K=I+1 TO N: LET D=A(K,I)/A(I,I)
120 FOR J=I+1 TO N: LET A(K,J)=A(K,J)-D*A(I,J):NEXT J:
    NEXT K
125 NEXT I
130 LET P=A(1,1): FOR I=2 TO N: LET P=P*A(I,I): NEXT I
140 CLS : PRINT AT 10,10; FLASH 1; "DETERMINANTUL
    MATRICII ESTE ="; FLASH 0;P
150 GO TO 64
```

10. Rezolvarea sistemelor de ecuatii liniare

10.1 Metoda bazată pe calculul matricei inverse

O metodă pentru rezolvarea unui sistem de n ecuaţii liniare cu n variabile, $Ax=b$,în cazul în care este compatibil şi determinat este bazată pe determinarea matricei A^{-1}, şi anume, soluţia sistemului este evident $x=A^{-1}$ b.

•P58

```
 10 REM ************************************************
 20 REM * Program BASIC pentru rezolvarea unui sistem de *
 30 REM * n ecuatii liniare cu n vriabile, Ax=b         *
 40 REM ************************************************
 50 INPUT "Numarul variabilelor = ";n
 60 INPUT "Numarul ecuatiilor = ";n1
 70 IF n=n1 THEN GO TO 120
 80 PRINT "Datele de intrare NU  corespund cerintelor"
 90 INPUT "Doriti reluarea ? (d/n) ";a$
100 IF a$="d" THEN GO TO 50
110 STOP
120 DIM b(n): DIM a(n,n)
130 PRINT : PRINT
140 PRINT "Introduceti coeficientii sistemului": PRINT
150 LET eps=0: LET k=0
160 FOR i=1 TO n
170 PRINT AT 20,5; FLASH 1;"Ecuatia ";i;":"
180 INPUT "Termenul liber = ";b(i)
190 FOR j=1 TO n
200 INPUT ("Coeficientul variabilei ";j;" = ");a(i,j)
210 IF a(i,j) <> 0 THEN LET eps=eps+ ABS a(i,j): LET k=k+1
220 NEXT j
230 NEXT i
240 LET eps=eps/k*1.E-8
250 REM 'eps' este pragul sub care un numar este considerat 0.
260 CLS
270 PRINT "Rezolvarea sistemului poate fi  realizata prin:"
280 PRINT
290 PRINT "1 -determinarea matricei inverse"
300 PRINT "2 -metoda Gauss-Jordan"
310 PRINT "3 -metoda Gauss-Seidel"
320 PRINT "0 -terminare program"
```

```
330 PRINT
340 INPUT "Formulati optiunea: ";op
350 IF op >= 0 AND op <= 3 THEN GO TO 370
360 PRINT "Optiune incorecta ! Repetati!": GO TO 340
370 IF op=1 THEN GO SUB 440
380 IF op=2 THEN GO SUB 1860
390 IF op=3 THEN GO SUB 2310
400 IF op=0 THEN CLS : STOP
410 PRINT : PRINT : PRINT
420 PRINT "Pentru continuare, apasati orice tasta !"
430 PAUSE 0: GO TO 260
440 REM ***********************************************
450 REM * Subprogram pentru calculul matricei inverse *
460 REM ***********************************************
470 REM Acest subprogram permite rezolvarea unui sistem
480 REM de ecuatii liniare prin determinarea matricei
490 REM inverse. In momentul apelarii coeficientii siste-
500 REM mului sint memorati in matricea A, termenii liberi
510 REM in vectorul B. Numarul ecuatiilor este N.
520 DIM g(n,n): DIM h(n,1)
530 LET irow=n: LET icol=n
540 LET jrow=n: LET jcol=1
550 LET inv=1
560 FOR i=1 TO n
570 FOR j=1 TO n
580 LET g(i,j)=a(i,j)
590 NEXT j: NEXT i
600 GO SUB 800
610 PRINT : PRINT
620 IF det=0 THEN GO TO 780
630 GO SUB 1440
640 FOR i=1 TO n
650 LET h(i,1)=b(i)
660 FOR j=1 TO n
670 LET g(i,j)=e(i,j+n)
680 NEXT j: NEXT i
690 LET icol=n
700 GO SUB 1690
710 REM f contine solutia sistemului.
720 REM Afisarea solutiei
730 PRINT "Solutia sistemului este:"
740 FOR i=1 TO n
750 PRINT "x(";i;") = ";f(i,1)
760 NEXT i
770 GO TO 790
780 PRINT "Matricea sistemului nu este inversabila!"
790 RETURN
```

```
800 REM **************************************************
810 REM * Subprogram BASIC pentru determinarea unei     *
820 REM * matrice superior triunghiulare prin metoda    *
830 REM * eliminarii Gaussiene. Matricea data este G    *
840 REM * avind irow linii si icol coloane. Matricea    *
850 REM * calculata de subprogram este submatricea      *
860 REM * lui E corespunzatoare coloanelor 1..icol.     *
870 REM **************************************************
880 IF irow=icol THEN GO SUB 910: GO TO 910
890 PRINT "Numarul de linii  <>  numarul de coloane."
900 RETURN
910 IF inv=1 THEN LET icol=2*icol
920 REM Linia 10 necesara numai pentru inversarea
930 REM unei matrice.
940 IF inv <> 1 THEN LET icol=icol+1
950 REM Linia 940 necesara pentru metoda Gauss-Jordan.
960 DIM e(irow,icol)
970 FOR i=1 TO irow
980 FOR j=1 TO irow
990 LET e(i,j)=g(i,j)
1000 IF inv <> 1 THEN GO TO 1040
1010 REM Linia 1030 este utilizata numai pentru
1020 REM determinarea inversei unei matrice.
1030 IF i=j THEN LET e(i,j+irow)=1
1040 NEXT j
1050 IF inv <> 1 THEN LET e(i,irow+1)=h(i,1)
1060 REM Linia 1050 este necesara pentru metoda
1070 REM Gauss-Jordan.
1080 NEXT i
1090 LET det=1
1100 FOR i=1 TO irow
1110 IF i <> irow THEN GO TO 1150
1120 IF e(i,i) <> 0 THEN GO TO 1250
1130 LET dt=0
1140 GO TO 1170
1150 IF e(i,i) <> 0 THEN GO TO 1190
1160 GO SUB 1270
1170 IF dt=0 AND inv <> 1 THEN LET d(i)=0
1180 IF dt=0 THEN LET det=0: GO TO 1250
1190 FOR j=i+1 TO irow
1200 LET xm=e(j,i)/e(i,i)
1210 FOR k=i TO icol
1220 LET e(j,k)=e(j,k)-xm*e(i,k)
1230 IF ABS e(j,k) < eps THEN LET e(j,k)=0
1240 NEXT k: NEXT j
1250 NEXT i
1260 RETURN
1270 REM *********************************
1280 REM * Subprogram pentru detectarea  *
1290 REM * determinantilor nuli          *
1300 REM *********************************
1310 REM Se incearca prin transformari elementare obtinerea
1320 REM de elemente diagonale nenule.
1330  REM Daca acest lucru nu este posibil, determinantul este 0.
```

```
1340 LET dt=1
1350 FOR j=i+1 TO irow
1360 IF e(j,i)=0 THEN GO TO 1410
1370 FOR k=1 TO icol
1380 LET e(i,k)=e(i,k)+e(j,k)
1390 IF ABS e(i,k)<eps THEN LET e(i,k)=0
1400 NEXT k: RETURN
1410 NEXT j
1420 LET dt=0
1430 RETURN
1440 REM **************************************************
1450 REM * Subprogram pentru determinarea formei inferior *
1460 REM * triunghiulare si transformarea in continuare   *
1470 REM * in matricea unitate. Datele de intrare sint    *
1480 REM * matricea E si dimensiunile sale irow si icol.  *
1490 REM **************************************************
1500 FOR k=1 TO irow
1510 IF k=irow THEN GO TO 1610
1520 LET ik=irow-k+1
1530 FOR i=1 TO ik-1
1540 IF inv=1 THEN GO TO 1560
1550 IF d(ik)=0 THEN GO TO 1610
1560 LET xm=e(i,ik)/e(ik,ik)
1570 FOR j=i+1 TO icol
1580 LET e(i,j)=e(i,j)-xm*e(ik,j)
1590 IF ABS e(i,j)<eps THEN LET e(i,j)=0
1600 NEXT j: NEXT i
1610 NEXT k
1620 IF det=0 THEN GO TO 1680
1630 FOR i=1 TO irow
1640 LET div=e(i,i)
1650 FOR j=1 TO icol
1660 LET e(i,j)=e(i,j)/div
1670 NEXT j: NEXT i
1680 RETURN
1690 REM ************************************************
1700 REM * Subprogram pentru calculul matricei F = G*H  *
1710 REM * Datele de intrare; G matrice cu irow linii,  *
1720 REM * icol coloane, H matrice cu jrow linii        *
1730 REM * si jcol coloane                              *
1740 REM ************************************************
1750 IF icol=jrow THEN GO TO 1780
1760 PRINT "Produsul matricelor nu se poate efectua!"
1770 GO TO 1850
1780 DIM f(irow,jcol)
1790 FOR i=1 TO irow
1800 FOR j=1 TO jcol
1810 LET f(i,j)=0
1820 FOR k=1 TO icol
1830 LET f(i,j)=f(i,j)+g(i,k)*h(k,j)
1840 NEXT k: NEXT j: NEXT i
1850 RETURN
```

10.2 Metoda Gauss-Jordan

Din punctul de vedere al volumului de calcul implicat, metoda Gauss-Jordan este preferabilă metodei bazate pe inversarea matricei coeficienţilor. Prin transformări elementare, matricea sistemului devine superior triunghiulară iar soluţia este :

$$x_{n-i} = \frac{b_{n-i}}{a_{n-i,n-i}} - \sum_{k=n-i+1}^{n} a_{n-i,k} \cdot x_k \qquad , i = 0, 1, \ldots, n-1$$

•P59

```
1860 REM ***********************************************
1870 REM * Subprogram BASIC pentru rezolvarea unui       *
1880 REM * sistem de ecuatii liniare prin metoda         *
1890 REM * Gauss-Jordan                                  *
1900 REM ***********************************************
1910 REM Datele de intrare sint:
1920 REM N = numarul ecuatiilor si a variabilelor,
1930 REM A = matricea coeficientilor sistemului,
1940 REM B = vectorul termenilor liberi.
1950 DIM g(n,n): DIM h(n,1): DIM d(n)
1960 LET irow=n: LET icol=n
1970 LET jrow=n: LET jcol=1
1980 FOR i=1 TO n
1990 FOR j=1 TO n
2000 LET g(i,j)=a(i,j): NEXT j
2010 LET h(i,1)=b(i): LET d(i)=1
2020 NEXT i
2030 LET inv=2
2040 GO SUB 800
2050 GO SUB 1440
2060 REM Rezultatul este ultima coloana din matricea G
2070 PRINT : PRINT
2080 PRINT "Rezultatele prin procedura Gauss-Jordan"
2090 PRINT
2100 REM Test pentru nedeterminare sau incopatibilitate
2110 LET stz=0
2120 FOR i=1 TO irow
2130 IF d(i) <> 0 THEN GO TO 2210
2140 LET stz=1
2150 FOR j=irow+1 TO icol
2160 IF e(i,j)=0 THEN GO TO 2190
2170 LET stz=2
2180 GO TO 2250
2190 NEXT j
2200 REM Iesirea normala semnifica sistem nedeterminat
2210 NEXT i
```

```
2220 IF stz=0 THEN GO TO 2270
2230 PRINT "Sistemul este compatibil nedeterminat"
2240 GO TO 2300
2250 PRINT "Sistemul este incompatibil"
2260 GO TO 2300
2270 FOR i=1 TO n
2280 PRINT "X(";i;")=";e(i,n+1)
2290 NEXT i
2300 RETURN
```

10.3 Metoda Gauss-Seidel

Metoda Gauss–Seidel este o metodă iterativă pentru rezolvarea sistemelor de ecuaţii liniare.

Fie sistemul:

$a_{11} x_1 + \ldots + a_{1n} x_n = b_1$
$a_{21} x_1 + \ldots + a_{2n} x_n = b_2$
....
$a_{n1} x_1 + \ldots + a_{nn} x_n = b_n$

Presupunem că ecuaţiile sînt astfel ordonate încît pentru toţi i, $1 \le i \le n$, coeficientul variabilei x_i din a i-a ecuaţie este cel mai mare in valoare absolută dintre coeficienţii acestei ecuaţii. Sistemul poate fi evident rescris astfel :

$x_1 = (1/a_{11})(b_1 - a_{12} x_2 - \ldots - a_{1n} x_n)$
$x_2 = (1/a_{22})(b_2 - a_{21} x_1 - \ldots - a_{2n} x_n)$
....
$x_n = (1/a_{nn})(b_n - a_{n1} x_1 - \ldots - a_{nn-1} x_{n-1})$

Considerăm aproximaţiile $\{x_k^* / 1 \le k \le n\}$ definite prin :

$x_1^* = b_1 / a_{11}$
$x_{2*} = (1/a_{22})(b_2 - a_{21} x_1^*)$
....
$x_n^* = (1/a_{nn})(bn - a_{n1} x_1^* - a_{n2} x_2^* - \ldots - a_{nn-1} x_{n-1}^*)$

Plecînd de la aceste aproximaţii, se rezolvă sistemul pentru fiecare variabilă. Soluţiile rezultate sînt în continuare utilizate pentru o nouă iteraţie, procesul iterativ continuînd pînă cînd două soluţii consecutive sînt suficient de apropiate.

Terminarea calculului se decide fie prin impunerea prealabilă a numărului de iteraţii, fie prin includerea unui test pe baza căruia se apreciază "cît de apropiate" sînt două soluţii succesive. Soluţia aproximativă a sistemului va fi rezultatul din ultima iteraţie.

•P60

```
2310 REM ************************************************
2320 REM * Subprogram pentru rezolvarea unui sistem      *
2330 REM * liniar prin metoda Gauss-Seidel.              *
2340 REM ************************************************
2350 REM Datele de intrare sint:
2360 REM A = matricea coefeicientilor sistemului,
2370 REM B = vectorul termenilor liberi,
2380 REM N = numarul de ecuatii si de variabile,
2390 REM EPSI = valoarea criteriului de oprire.
2400 DIM d(n)
2410 INPUT "Valoarea criteriului de oprire = ";epsi
2420 PRINT : PRINT
2430  INPUT "Doriti rezultatele la fiecare iteratie? (d/n)";a$
2440 REM Determinarea coeficientului maxim
2450 REM din fiecare ecuatie
2460 FOR i=1 TO n
2470 LET max= ABS a(i,i): LET d(i)=i
2480 FOR j=1 TO n
2490 IF ABS a(i,j) <= max THEN GO TO 2520
2500 LET max= ABS a(i,j)
2510 LET d(i)=j
2520 NEXT j: NEXT i
2530 REM Permutarea liniilor matricii A in speranta ca
2540 REM componentele maxime in valoare absoluta sa fie
2550 REM pe diagonala principala
2560 LET test=1
2570 FOR i=1 TO n-1
2580 IF d(i)=i THEN GO TO 2740
2590 FOR k=1 TO n
2600 IF d(k) <> i THEN GO TO 2630
2610 LET jj=k
2620 GO TO 2650
2630 NEXT k
2640 LET test=0: GO TO 2740
2650 FOR j=1 TO n
2660 LET d=a(jj,j)
2670 LET a(jj,j)=a(i,j)
2680 LET a(i,j)=d
2690 NEXT j
2700 LET d=b(i)
2710 LET b(i)=b(jj)
2720 LET b(jj)=d
2730 LET d(jj)=d(i)
2740 NEXT i
2750 IF test THEN GO TO 2800
2760 PRINT : PRINT
2770 PRINT "Matricea sistemului NU este diagonal dominanta."
2780 PRINT "Metoda ar putea sa NU convearga !": PRINT
```

```
2790 REM Calculele pentru metoda Gauss-Seidel
2800 REM Aproximatiile initiale
2810 DIM x(n)
2820 FOR j=1 TO n
2830 LET sum=0
2840 IF j=1 THEN GO TO 2880
2850 FOR i=1 TO j-1
2860 LET sum=sum+x(i)
2870 NEXT i
2880 LET x(j)=(b(j)-sum)/a(j,j)
2890 NEXT j
2900 REM Rezolvarea prin metoda iterativa
2910 LET ik=0
2920 LET ks=1
2930 LET ik=ik+1
2940 IF ik>100 THEN GO TO 3210
2950 FOR i=1 TO n
2960 LET sum=0
2970 FOR j=1 TO n
2980 IF i=j THEN GO TO 3000
2990 LET sum=sum+a(i,j)*x(j)
3000 NEXT j
3010 LET xx=x(i)
3020 LET x(i)=(b(i)-sum)/a(i,i)
3030 IF ABS (xx-x(i))>epsi THEN LET ks=2
3040 NEXT i
3050 IF a$="n" THEN GO TO 3120
3060 PRINT : PRINT
3070 PRINT "La iteratia ";ik
3080 FOR i=1 TO n
3090 PRINT TAB 8;"X(";i;")=";x(i)
3100 NEXT i
3110 INPUT "Apasati orice tasta pentru continuare";b$
3120 IF ks=2 THEN GO TO 2920
3130 REM Conditia de terminare indeplinita! ---> afisarea
3140 REM rezultatelor
3150 PRINT "Aproximatia solutiei prin metoda Gauss-Seidel"
3160 PRINT "pentru prag eroare EPSI=";epsi: PRINT
3170 FOR i=1 TO n
3180 PRINT "X(";i;")=";x(i)
3190 NEXT i
3200 GO TO 3230
3210 PRINT "Dupa 100 iteratii aproximatie nesatisfacatoare!"
3220 PRINT "Probabil sirul aproximatiilor este divergent"
3230 RETURN
```

11. Rezolvarea ecuaţiilor neliniare

11.1 Schema lui Horner

•P61

```
 5 REM SCHEMA LUI HORNER
 20 INPUT "INDICATI GRADUL POLINOMULUI ";N
 30 DIM C (N+1)
 40 FOR I=1 TO N+1
 50 INPUT ("INTRODUCETI COEFICIENTUL C (";I;")= ");C(I)
 60 NEXT I
 80 INPUT "INDICATI VALOAREA PENTRU A= ";A
 85 PRINT FLASH 1; "POLINOMUL DAT ESTE": PRINT: FOR
    I=1 TO N+1: PRINT C (I);" ";: NEXT I
 86 PRINT : PRINT : PRINT FLASH 1; "VALOAREA LUI A ESTE
    = "; FLASH 0 ;A
 87 FOR I=1 TO 10: BEEP 3*RND, I*RND: NEXT I: CLS
 88 PRINT AT 1,0; FLASH 1; "COEFICIENTII CÎTULUI SINT"
110 FOR I=2 to m+1
120 LET C (I)=C (I-1)*A+C(I)
140 NEXT I
145 PRINT : FOR I=1 TO N: PRINT C(I);" ";:
    NEXT I: PRINT
150 PRINT FLASH 1; "RESTUL ESTE "; FLASH 0;C (N+1)
```

11.2 Aproximarea soluţiei unei ecuaţii f(x)=0

Programul care urmează calculează o rădăcină a ecuaţiei f(x)=0, unde f este o funcţie continuă de o variabilă reală x.

Metoda utilizată este o combinaţie între metoda înjumătăţirii intervalului şi cea a secantei.

Pentru execuţia programului trebuie definită funcţia f (.) şi intervalul [a,b] în care avem f(a)*f(b)<0. In plus, mai sînt cerute şi toleranţele: relerr = eroarea relativă abserr= eroarea absolută.

Funcţia f(x) se defineşte ca un subprogram funcţie cu numărul de ordine 10, de exemplu: 10 DEF FN f(x)=3*x+SIN x

Acest program este adaptat după lucrarea [19].

•P62

```
  1 REM ***************************************
  2 REM * Program BASIC pentru determinarea   *
  3 REM * unei radacini a ecuatiei f(x) = 0   *
  4 REM ***************************************
 10 REM ---  DEF FN f(x)=.... ---
 20 CLS : PRINT "Definiti capetele intervalului:"
 30 INPUT "a = ";b: INPUT "b = ";c
 40 PRINT : PRINT TAB 8;"[";b;",";c;"]"
 50 PRINT : PRINT "Precizati tolerantele:"
 60 INPUT "0  <=  relerr = ";relerr
 70 INPUT "0 < abserr = ";abserr: PRINT
 80 PRINT TAB 8;"relerr = ";relerr
 90 PRINT TAB 8;"abserr = ";abserr
100 PRINT AT 15,6; FLASH 1;" R E Z O L V A R E "
110 LET flag=10: LET max=500: LET u=1.E-8: DIM e$(2,5)
120 LET e$(1)="minim": LET e$(2)="maxim"
130 GO SUB 400
140 IF flag<0 THEN LET ft= FN f(t): GO TO 130
150 PRINT AT 15,6; TAB 25: LET k=1: IF SGN ft=-1 THEN LET k=2
160 IF flag>2 THEN GO TO 190
170 PRINT AT 12,0;"Radacina gasita este:"
180 PRINT : PRINT TAB 8;"x = ";b: GO TO 340
190 IF flag <> 3 THEN GO TO 250
200 PRINT AT 12,0;"Valoarea  ABS f(";b;")"
210 PRINT "e mai mare decit MAX{f(a),f(b)}."
220 PRINT "Exista posibilitatea unei"
230 PRINT "discontinuitati in vecinatatea"
240 PRINT "punctului  x = ";b;" ?!": GO TO 340
250 IF flag <> 4 THEN GO TO 310
260 PRINT AT 12,0;"In intervalul considerat NU s-a"
270 PRINT "determinat o radacina de ordin"
280 PRINT "impar. x = ";b
290 PRINT "este un punct de ";e$(k);" local cu"
300 PRINT "f(";b;") = ";ft: GO TO 340
310 IF flag <> 5 THEN GO TO 340
320 PRINT AT 12,0;"S-au efectuat prea multe"
330 PRINT "evaluari ale functiei. (max=500)"
340 PRINT : PRINT : PRINT "Doriti definirea unui interval"
350 PRINT "nou si/sau schimbarea"'"olerantelor ? (y/n)"
360 LET a$= INKEY$
370 IF a$="y" THEN GO TO 20
380 IF a$="n" THEN STOP
390 GO TO 360
400 IF flag >= 0 THEN GO TO 450
410 LET flag= ABS flag
420 IF flag=1 THEN GO TO 500
430 IF flag=2 THEN GO TO 520
440 IF flag=3 THEN GO TO 790
450 LET re=u: IF u<relerr THEN LET re=relerr
460 LET ae=abserr: LET ic=0
```

```
470 LET acbs= ABS (b-c)
480 LET a=c: LET t=a
490 LET flag=-1: RETURN
500 LET fa=ft: LET t=b
510 LET flag=-2: RETURN
520 LET fb=ft: LET fc=fa
530 LET kount=2
540 LET fx= ABS fb: IF fx< ABS fc THEN LET fx= ABS fc
550 IF ABS fc >= ABS fb THEN GO TO 590
560 LET a=b: LET fa=fb
570 LET b=c: LET fb=fc
580 LET c=a: LET fc=fa
590 LET cmb=(c-b)/2
600 LET acmb= ABS cmb
610 LET tol=re* ABS b+ae
620 IF acmb <= tol THEN GO TO 840
630 IF kount >= max THEN GO TO 900
640 LET p=(b-a)*fb
650 LET q=fa-fb
660 IF p >= 0 THEN GO TO 680
670 LET p=-p: LET q=-q
680 LET a=b: LET fa=fb
690 LET ic=ic+1
700 IF ic<4 THEN GO TO 730
710 IF 8*acmb>acbs THEN GO TO 770
720 LET ic=0: LET acbs=acmb
730 IF p> ABS q*tol THEN GO TO 750
740 LET b=b+tol* SGN cmb: GO TO 780
750 IF p >= cmb*q THEN GO TO 770
760 LET b=b+p/q: GO TO 780
770 LET b=(c+b)/2
780 LET t=b: LET flag=-3: RETURN
790 LET fb=ft
800 IF fb=0 THEN GO TO 870
810 LET kount=kount+1
820 IF SGN fb <> SGN fc THEN GO TO 550
830 LET c=a: LET fc=fa: GO TO 550
840 IF SGN fb= SGN fc THEN GO TO 890
850 IF ABS fb>fx THEN GO TO 880
860 LET flag=1: RETURN
870 LET flag=2: RETURN
880 LET flag=3: RETURN
890 LET flag=4: RETURN
900 LET flag=5: RETURN
```

11.3 Metoda Newton Raphson

Metoda Newton-Raphson face parte din clasa metodelor iterative care permit aproximarea soluţiilor ecuaţiilor neliniare. Subprogramul prezentat în cadrul acestei secţiuni realizează aplicarea metodei Newton-Raphson la rezolvarea ecuaţiilor

polinomiale. Fie f(x) = 0 o ecuaţie polinomială, f polinom de gradul n cu coeficienţi reali. Metoda constă în generarea termenilor şirului $(x_n)_{n \geq 1}$; unde x_1 este aproximaţia iniţială pentru o soluţie a ecuaţiei; $x_{k+1} = x_k - f(x_k)/f'(x_k)$; $k \geq 1$, f' derivata funcţiei f. Decizia asupra terminării calculului este luată pe baza unui test asupra modulului valorii $f(x_k)$.

•P63

```
100 REM *********************************************
105 REM * Metoda Newton-Raphson pentru determinarea        *
110 REM * solutiilor reale ale unei ecuatii polinomiale *
115 REM * Datele de intrare :                              *
120 REM *   - gradul polinomului -variabila JD             *
125 REM *   - coeficientii polinomului-vectorul F          *
130 REM *   - aproximatia initiala-variabila X             *
135 REM *   - criteriul de oprire-variabila Z              *
140 REM * Subprogramul pentru metoda Newton-Raphson        *
145 REM * linia de program 2600                            *
150 REM *********************************************
210 INPUT "Gradul polinomului ";jd
220 DIM f(jd+1)
230 INPUT "Valoarea criteriu-test de oprire ";z
240 FOR i=1 TO jd+1
250 PRINT "Coeficientul lui x la puterea ";i-1
260 INPUT f(i)
270 NEXT i
280 INPUT "Aproximatia initiala ";x
290 GOSUB 2600
300 INPUT "Doriti reluarea pentru alta valoare initiala
    ?(d/n)";a$
310 IF a$="d" THEN GOTO 280
320 STOP
2600 REM ******************************************
2610 REM *       Subprogramul Newton -Raphson          *
2620 REM ******************************************
2630 LET jj=1:LET x1=x
2640 LET sum=0:LET psum =0
2645 LET xp=1:LET xp1=1
2650 FOR i=1 TO jd+1
2660 LET sum=sum+f(i)*xp
2670 IF i=0 THEN GOTO 2690
2680 LET psum=psum+f(i)*i*xp1
2690 LET xp1=xp:LET xp=xp*x
2700 NEXT i
2710 IF ABS(sum)<=z THEN GOTO 2850
2720 IF psum=0 THEN GOTO 2830
2730 LET x=x-sum/psum
2740 LET jj=jj+1
2750 IF jj>100 THEN GOTO 2870
2800 GOTO 2640
```

```
2830 PRINT "Pentru aproximatia initiala ";x1;
     "panta este nula"
2840 GOTO 2880
2850 PRINT "Pentru aproximatia initiala ";x1;
     "radacina este";x
2860 GOTO 2880
2870 PRINT "Radacina nu a fost determinata dupa
     100 iteratii"
2880 RETURN
```

12. Integrarea numerică

2.1 Metoda RUNGE-KUTTA

Programul realizează integrarea unui sistem de ecuaţii diferenţiale de ordinul 1 de forma :

dY /dx = F(x,Y)

utilizînd metoda RUNGE-KURTTA de ordinul 4.

Notaţiile au următoarele semnificaţii :

x - variabilă independentă

Y - funcţia (vectorială) de integrat. $Y : R \rightarrow R^n$

F - funcţia care defineşte sistemul. $F : R^{n+1} \rightarrow R^n$

Sistemul de ecuaţii trebuie definit ca subprogram, liniile de program avînd numerele de ordine începînd cu 10 pînă la 99 cel mult.

De exemplu :

25 LET d(3) = Y(2) + x - Y(1)

ATENŢIE!

Funcţia de integrat Y şi derivata ei d sînt vectori! (eventual de dimensiune 1).

După introducerea instrucţiunilor pentru definirea sistemului de ecuaţii diferenţiale comanda RUN 100 determină lansarea în execuţie a programului, acesta solicitînd în mod convenţional elementele necesare integrării: număr de ecuaţii, intervalul de integrare, valoarea funcţiei în punctul iniţial, precizia de integrare.

•P64

```
1 REM ***************************************************
2 REM * Program BASIC pentru integrarea unui sistem de *
3 REM * ecuatii diferentiale de ordinul 1 prin metoda  *
4 REM *             R U N G E - K U T T A              *
5 REM ***************************************************
6 PRINT "Introduceti sistemul de ecuatii,": PRINT
7 PRINT "dupa care executati: "; FLASH 1;"RUN 100"
```

```
  8 STOP
  9 REM --- Sistemul de ecuatii ---
 10 REM ... LET d(.) = ......
 99 RETURN
100 CLS : PRINT "Introduceti numarul de ecuatii:"
110 INPUT "nrec=";nrec: PRINT TAB 8;"nrec=";nrec
120 DIM y(nrec): DIM d(nrec): DIM f(3,nrec): DIM r(nrec)
130 PRINT : PRINT "Precizati intervalul de integrare:"
140 INPUT "x-initial=";x0: INPUT "x-final=";x1
150 PRINT AT 4,10;"[";x0;",";x1;"]": PRINT
160 PRINT "Introduceti valorile functiei y in punctul initial:"
170 FOR i=1 TO nrec
180 INPUT "y=";y(i): PRINT TAB 4;"y";i;"(";x0;")=";y(i)
190 NEXT i: PRINT
200 PRINT "Precizia de integrare la capatulintervalului:"
210 INPUT "eps=";eps: PRINT TAB 8;"eps =";eps
220 PRINT : PRINT : PRINT TAB 6; FLASH 1;" R E Z O L V A R E "
230 IF x0=x1 THEN GO TO 510
240 LET i=1
250 LET h= ABS (x1-x0)/i
260 IF h<eps THEN GO TO 280
270 LET i=i+1: GO TO 250
280 LET h=h* SGN (x1-x0)
290 LET hs2=h/2
300 FOR s=x0 TO x1 STEP h
310 LET x=s: GO SUB 10
320 FOR i=1 TO nrec
330 LET r(i)=y(i)
340 LET f(1,i)=d(i)
350 LET y(i)=r(i)+d(i)*hs2
360 NEXT i
370 LET x=s+hs2: GO SUB 10
380 FOR i=1 TO nrec
390 LET f(2,i)=d(i)
400 LET y(i)=r(i)+d(i)*hs2
410 NEXT i
420 GO SUB 10
430 FOR i=1 TO nrec
440 LET f(3,i)=d(i)
450 LET y(i)=r(i)+d(i)*h
460 NEXT i
470 LET x=s+h: GO SUB 10
480 FOR i=1 TO nrec
490 LET y(i)=r(i)+h*(f(1,i)+d(i)+ 2*(f(2,i)+f(3,i)))/6: NEXT i
500 PRINT AT 20,0;"S-a evaluat y pt. x=";s: NEXT s
510 CLS : PRINT "Valorile functiei in punctul final:": PRINT
520 FOR i=1 TO nrec
530 PRINT TAB 4;"y";i;"(";x1;") = ";y(i): NEXT i
```

12.2 Metoda Adams

Programul realizează integrarea unui sistem de ecuaţii diferenţiale de ordinul 1 de forma :

$$dY/dx = F(x,Y)$$

utilizînd o metodă de tip ADAMS predictor-corector cu pas şi grad de interpolare variabile. Programul este adaptat după lucrarea [20].

Semnificaţiile corespunzătoare notaţiilor sînt:

x - variabila indepedendentă

Y - funcţia de integrat. $Y : R \rightarrow R^n$

F - funcţia care defineşta sistemul. $F : R^{n+1} \rightarrow R^n$

Sistemul de ecuaţii trebuie definit ca subprogram, liniile de program corespunzătoare acestuia avînd numerele de ordine de la 10 la 99 cel mult.

Evaluarea derivatelor (dY/dx) se va realiza în vectorul d (.).

De exemplu:

25 LET d(3) = Y(2) + x - Y(1)

ATENŢIE!

Funcţia de integrat Y şi derivata ei d sînt vectori! (eventual de dimensiune 1).

După introducerea instrucţiunilor necesare definirii sistemului de ecuaţii diferenţiale se solicită prin comanda RUN 100 lasarea în execuţie a programului, în faza de execuţie programul solicitînd în mod conversaţional elementele necesare integrării: numărul de ecuaţii, intervalul de integrare, valoarea funcţiei în punctul iniţial, eroarea relativă, eroarea absolută, opţiunea asupra posibilităţii depăşirii capătului final al intervalului.

●P65

```
  1 REM ****************************************************
  2 REM * Program BASIC pentru integrarea unui sistem de   *
  3 REM * ecuatii diferentiale de ordinul 1, prin metoda   *
  4 REM *           ADAMS - predictor - corector.          *
  5 REM ****************************************************
  6 PRINT "Introduceti sistemul de ecuatii,": PRINT
  7 PRINT "dupa care executati:  "; FLASH 1;"RUN 100"
  8 STOP
  9 REM --- Sistemul de ecuatii ---
 10 REM  ... LET d(.) = ......
 99 RETURN
100 CLS : PRINT "Introduceti numarul de ecuatii:"
```

```
110 INPUT "nrec=";nrec: PRINT TAB 8;"nrec=";nrec
120 DIM y(nrec): DIM t(nrec): DIM h(nrec,16)
130 DIM x(nrec): DIM d(nrec): DIM s(12): DIM a(12)
140 DIM b(12): DIM c(13): DIM w(12): DIM v(12): DIM g(13)
150 DIM r(13): DIM z(13): DIM o(13): DIM e(13): DIM p(nrec)
160 DATA 2,4,8,16,32,64,128,256,512,1024,2048,4096,8192
170 DATA .5,.0833,.0417,.0264,.0188,.0143,.0114
180 DATA .00936,.00789,.00679,.00592,.00524,.00468
190 DATA 1,.5,1,1E-9,2E-9,1,500
200 FOR i=1 TO 13: READ z(i): NEXT i
210 FOR i=1 TO 13: READ r(i): NEXT i
220 READ g(1),g(2),c(1),twou,fouru,o(1),maxnum: RESTORE 160
230 PRINT : PRINT "Precizati intervalul de integrare:"
240 INPUT "x-initial=";t: INPUT "x-final=";tout
250 PRINT AT 4,10;"[";t;",";tout;"]": PRINT
260 PRINT "Introduceti valorile functiei y in punctul initial:"
270 FOR i=1 TO nrec
280 INPUT ("y";i;"(";t;")=");y(i)
290 PRINT TAB 4;"y";i;"(";t;")=";y(i)
300 NEXT i
310 PRINT : PRINT "Eroarea relativa:"
320 INPUT "0 <= relerr=";relerr: PRINT TAB 4;"relerr = ";relerr
330 PRINT : PRINT "Eroarea absoluta:"
340 INPUT "0<abserr=";abserr: PRINT TAB 4;"abserr = ";abserr
350 PRINT : PRINT "Integrarea poate depasi capatul"
360 PRINT "intervalului(x-final) ? (y/n)"
370 LET a$= INKEY$
380 IF a$="y" THEN LET flag=1: GO TO 410
390 IF a$="n" THEN LET flag=-1: GO TO 410
400 GO TO 370
410 PRINT : PRINT TAB 6; FLASH 1;" R E Z O L V A R E "
420 PRINT : PRINT : PRINT
430 GO SUB 760
440 CLS
450 IF ( ABS flag <> 2) THEN GO TO 490
460 PRINT "Valorile functiei in capatul intervalului sint:"
470 FOR i=1 TO nrec: PRINT "    y";i;"(";tout;")=";y(i)
480 NEXT i: GO TO 660
490 IF ( ABS flag <> 3) THEN GO TO 550
500 PRINT "Integrarea n-a atins x-final"
510 PRINT "datorita tolerantelor prea mici."
520 PRINT "'relerr' si/sau 'abserr' sint"
530 PRINT "modificate corespunzator preci-"
540 PRINT "ziei calculatorului.": PAUSE 500: GO TO 670
550 IF ( ABS flag <> 4) THEN GO TO 600
560 PRINT "Integrarea n-a atins x-final"
570 PRINT "datorita depasirii numarului"
580 PRINT "maxim de iteratii (maxnum=500)"
590 PAUSE 500: GO TO 670
600 IF ( ABS flag <> 5) THEN GO TO 640
610 PRINT "Integrarea n-a atins x-final"
620 PRINT "deoarece ecuatiile par a fi"
```

```
630 PRINT "rigide (stiff).": PAUSE 500: GO TO 670
640 IF ( ABS flag  <>  6) THEN GO TO 660
650 PRINT "Date de intrare eronate!": PAUSE 500: GO TO 100
660 IF isn<0 THEN STOP
670 PRINT : PRINT "Doriti continuarea integrarii ? (y/n)"
680 LET a$= INKEY$
690 IF (a$ <> "y" OR flag <> 2) THEN GO TO 730
700 INPUT "x-final=";tout: PRINT
710 PRINT "Noul interval de integrare:"
720 PRINT TAB 8;"[";t;",";tout;"]": GO TO 410
730 IF ( ABS flag >= 3 AND a$="y" AND ABS flag <= 5) THEN GO TO 250
740 IF a$="n" THEN STOP
750 GO TO 680
760 LET isn= SGN flag: LET flag= ABS flag
770 IF t=tout THEN LET flag=2*isn: RETURN
780 IF (relerr<0 OR abserr <= 0) THEN LET flag=6*isn:  RETURN
790 LET eps=relerr: IF (eps<abserr) THEN LET eps=abserr
800 IF flag=1 THEN GO TO 820
810 IF t <> told THEN LET flag=6*isn: RETURN
820 LET del=tout-t: LET absdel= ABS del: LET tend=t+10*del
830 IF isn<0 THEN LET tend=tout
840 LET nostep=0: LET kle4=0: LET stiff=0
850 LET releps=relerr/eps: LET abseps=abserr/eps
860 IF flag=1 THEN GO TO 890
870 IF isnold<0 THEN GO TO 890
880 IF delsgn*del>0 THEN GO TO 930
890 LET start=1: LET x=t
900 LET delsgn= SGN del
910 LET h= ABS (tout-x): IF h<fouru* ABS x THEN LET h=fouru* ABS x
920 LET h=h* SGN (tout-x)
930 IF ABS (x-t)<absdel THEN GO TO 970
940 GO SUB 2940: REM (INTRP)
950 LET flag=2: LET t=tout: LET told=t: LET isnold=isn
960 RETURN
970 IF (isn>0 OR ABS (tout-x) >= fouru* ABS (x)) THEN GO TO 1020
980 LET h=tout-x
990 GO SUB 10
1000 FOR l=1 TO nrec: LET y(l)= y(l)+h*d(l): NEXT l
1010 LET flag=2: LET t=tout: LET told=t: LET isnold=isn: RETURN
1020 IF nostep<maxnum THEN GO TO 1050
1030 LET flag=isn*4: IF stiff THEN LET flag=isn*5
1040 LET t=x: LET told=t: LET isnold=1: RETURN
1050 LET hh= ABS h: IF hh> ABS (tend-x) THEN LET hh= ABS (tend-x)
1060 LET h=hh* SGN h
1070 FOR l=1 TO nrec: LET t(l)= releps* ABS y(l)+abseps: NEXT l
1080 GO SUB 1190: REM (STEP)
1090 IF ( NOT crash) THEN GO TO  1130
1100 LET flag=isn*3: LET relerr=eps*releps
1110 LET abserr=eps*abseps
1120 LET t=x: LET told=t: LET isnold=1: RETURN
1130 LET nostep=nostep+1: LET kle4=kle4+1
1140 PRINT AT 20,0;"Nr.pas=";nostep
1150 PRINT AT 21,0;"S-a evaluat y in x=";x
```

```
1160 IF kold>4 THEN LET kle4=0
1170 IF kle4 >= 50 THEN LET stiff=1
1180 GO TO 930
1190 REM   S T E P
1200 LET crash=1
1210 IF ( ABS h >= fouru* ABS x) THEN GO TO 1240
1220 LET h=fouru* ABS x* SGN h
1230 RETURN
1240 LET p5eps=eps/2
1250 LET round=0: FOR l=1 TO nrec
1260 LET temp=y(l)/t(l): LET round=round+temp*temp: NEXT l
1270 LET round=twou* SQR round
1280 IF p5eps >= round THEN GO TO 1310
1290 LET eps=2*round*(1+fouru)
1300 RETURN
1310 LET crash=0
1320 IF ( NOT start) THEN GO TO 1450
1330 GO SUB 10
1340 LET sum=0
1350 FOR l=1 TO nrec: LET h(l,1)=d(l): LET h(l,2)=0
1360 LET temp=d(l)/t(l): LET sum=sum+temp*temp: NEXT l
1370 LET sum= SQR sum: LET absh= ABS h
1380 IF eps<16*sum*h*h THEN LET absh=.25* SQR (eps/sum)
1390 IF ABS h<fouru* ABS x THEN LET h=fouru* ABS x* SGN h
1400 LET hold=0: LET k=1: LET kold=0: LET start=0
1410 LET phase1=1: LET nornd=1
1420 IF p5eps>100*round THEN GO TO 1450
1430 LET nornd=0
1440 FOR l=1 TO nrec: LET h(l,15)=0: NEXT l
1450 LET ifail=0:
1460 LET kp1=k+1: LET kp2=k+2
1470 LET km1=k-1: LET km2=k-2
1480 IF h <> hold THEN LET ns=0
1490 LET ns=ns+1: IF ns>kold+1 THEN LET ns=kold+1
1500 LET nsp1=ns+1: IF k<ns THEN GO TO 1780
1510 LET b(ns)=1: LET a(ns)=1/ns: LET temp1=h*ns: LET c(nsp1)=1
1520 IF k<nsp1 THEN GO TO 1570
1530 FOR i=nsp1 TO k: LET im1=i-1: LET temp2=s(im1)
1540 LET s(im1)=temp1: LET b(i)=b(im1)*temp1/temp2
1550 LET temp1=temp2+h: LET a(i)=h/temp1
1560 LET c(i+1)=i*a(i)*c(i): NEXT i
1570 LET s(k)=temp1
1580 IF ns>1 THEN GO TO 1620
1590 FOR i=1 TO k: LET temp3=i*(i+1)
1600 LET v(i)=1/temp3:  LET w(i)=v(i): NEXT i
1610 GO TO 1720
1620 IF k <= kold THEN GO TO 1670
1630 LET temp4=k*kp1: LET v(k)=1/temp4: LET nsm2=ns-2
1640 IF nsm2<1 THEN GO TO 1670
1650 FOR j=1 TO nsm2: LET i=k-j
1660 LET v(i)=v(i)-a(j+1)*v(i+1): NEXT j
1670 LET limit1=kp1-ns
1680 LET temp5=a(ns)
```

```
1690 FOR i=1 TO limit1
1700 LET v(i)=v(i)-temp5*v(i+1): LET w(i)=v(i): NEXT i
1710 LET g(nsp1)=w(1)
1720 LET nsp2=ns+2
1730 IF kp1<nsp2 THEN GO TO 1780
1740 FOR i=nsp2 TO kp1
1750 LET limit2=kp2-i: LET temp6=a(i-1)
1760 FOR j=1 TO limit2: LET w(j)=w(j)-temp6*w(j+1): NEXT j
1770 LET g(i)=w(1): NEXT i
1780 IF k<nsp1 THEN GO TO 1830
1790 FOR i=nsp1 TO k
1800 LET temp1=b(i)
1810 FOR l=1 TO nrec: LET h(l,i)=temp1*h(l,i): NEXT l
1820 NEXT i
1830 FOR l=1 TO nrec
1840 LET h(l,kp2)=h(l,kp1)
1850 LET h(l,kp1)=0
1860 LET p(l)=0: NEXT l
1870 FOR j=1 TO k
1880 LET i=kp1-j: LET ip1=i+1: LET temp2=g(i)
1890 FOR l=1 TO nrec
1900 LET p(l)=p(l)+temp2*h(l,i): LET h(l,i)=h(l,i)+h(l,ip1)
1910 NEXT l
1920 NEXT j
1930 IF nornd THEN GO TO 1980
1940 FOR l=1 TO nrec
1950 LET tau=h*p(l)-h(l,15): LET p(l)=y(l)+tau: LET h(l,16)=0
1960 NEXT l
1970 GO TO 2000
1980 FOR l=1 TO nrec
1990 LET p(l)=y(l)+h*p(l): NEXT l
2000 LET xold=x: LET x=x+h: LET absh= ABS h
2010 FOR l=1 TO nrec: LET x(l)=y(l): LET y(l)=p(l): NEXT l
2020 GO SUB 10
2030 FOR l=1 TO nrec: LET y(l)=x(l): NEXT l
2040 LET erkm2=0: LET erkm1=0: LET erk=0
2050 FOR l=1 TO nrec
2060 LET temp3=1/t(l)
2070 LET temp4=d(l)-h(l,1)
2080 IF km2<0 THEN GO TO 2120
2090 IF km2=0 THEN GO TO 2110
2100 LET temp=(h(l,km1)+temp4)*temp3: LET erkm2=erkm2+temp*temp
2110 LET temp=(h(l,k)+temp4)*temp3: LET rekm1=erkm1+temp*temp
2120 LET temp=temp4*temp3: LET erk=erk+temp*temp
2130 NEXT l
2140 IF km2<0 THEN GO TO 2180
2150 IF km2=0 THEN GO TO 2170
2160 LET erkm2=absh*c(km1)*r(km2)* SQR erkm2
2170 LET erkm1=absh*c(k)*r(km1)* SQR erkm1
2180 LET temp5=absh* SQR erk: LET err=temp5*(g(k)-g(kp1))
2190 LET erk=temp5*c(kp1)*r(k): LET knew=k
2200 IF km2<0 THEN GO TO 2270
```

```
2210 IF km2=0 THEN GO TO 2260
2220 LET temp1=erkm1
2230 IF erkm1<erkm2 THEN LET temp1=erkm2
2240 IF temp1 <= erk THEN LET knew=km1
2250 GO TO 2270
2260 IF erkm1 <= .5*erk THEN LET knew=km1
2270 IF err <= eps THEN GO TO 2480
2280 LET phase1=0: LET x=xold
2290 FOR i=1 TO k
2300 LET temp1=1/b(i)
2310 LET ip1=i+1
2320 FOR l=1 TO nrec: LET h(l,i)=temp1*(h(l,i)-h(l,ip1)): NEXT l
2330 NEXT i
2340 IF k<2 THEN GO TO 2360
2350 FOR i=2 TO k: LET s(i-1)=s(i)-h: NEXT i
2360 LET ifail=ifail+1
2370 LET temp2=.5
2380 IF ifail<3 THEN GO TO 2420
2390 IF ifail=3 THEN GO TO 2410
2400 IF p5eps<.25*erk THEN LET temp2= SQR (p5eps/erk)
2410 LET knew=1
2420 LET h=temp2*h: LET k=knew
2430 IF ABS h >= fouru* ABS x THEN GO TO 2470
2440 LET crash=1
2450 LET h=fouru* ABS x* SGN h
2460 LET eps=eps+eps: RETURN
2470 GO TO 1460
2480 LET kold=k: LET hold=h
2490 LET temp1=h*g(kp1)
2500 IF nornd THEN GO TO 2540
2510 FOR l=1 TO nrec: LET rho=temp1*(d(l)-h(l,1))-h(l,16)
2520 LET y(l)=p(l)+rho: LET h(l,15)=0: NEXT l
2530 GO TO 2560
2540 FOR l=1 TO nrec
2550 LET y(l)=p(l)+temp1*(d(l)-h(l,1)): NEXT l
2560 GO SUB 10
2570 FOR l=1 TO nrec
2580 LET h(l,kp1)=d(l)-h(l,1)
2590 LET h(l,kp2)=h(l,kp1)-h(l,kp2): NEXT l
2600 FOR i=1 TO k
2610 FOR l=1 TO nrec
2620 LET h(l,i)=h(l,i)+h(l,kp1): NEXT l
2630 NEXT i
2640 LET erkp1=0: IF (knew=km1 OR k=12) THEN LET phase1=0
2650 IF phase1 THEN GO TO 2760
2660 IF knew=km1 THEN GO TO 2790
2670 IF kp1>ns THEN GO TO 2810
2680 FOR l=1 TO nrec: LET temp=h(l,kp2)/t(l)
2690 LET erkp1=erkp1+temp*temp: NEXT l
2700 LET erkp1=absh*r(kp1)* SQR erkp1
2710 IF k>1 THEN GO TO 2740
2720 IF erkp1 >= .5*erk THEN GO TO 2810
```

```
2730 GO TO 2760
2740 IF (erkml <= erk AND erkml <= erkp1) THEN GO TO 2790
2750 IF (erkp1 >= erk OR k=12) THEN GO TO 2810
2760 LET k=kp1
2770 LET erk=erkp1
2780 GO TO 2810
2790 LET k=km1
2800 LET erk=erkm1
2810 LET hnew=h+h
2820 IF phase1 THEN GO TO 2930
2830 IF p5eps >= erk*z(k+1) THEN GO TO 2930
2840 LET hnew=h
2850 IF p5eps >= erk THEN GO TO 2930
2860 LET temp2=k+1
2870 LET r=(p5eps/erk)^(1/temp2)
2880 LET hnew=.9: IF .9>r THEN LET hnew=r
2890 IF .5>hnew THEN LET hnew=.5
2900 LET hnew=absh*hnew
2910 IF hnew<fouru* ABS x THEN LET hnew=fouru* ABS x
2920 LET hnew=hnew* SGN h
2930 LET h=hnew: RETURN
2940 REM   I N T R P
2950 LET hi=tout-x
2960 LET ki=kold+1
2970 LET kip1=ki+1
2980 FOR i=1 TO ki: LET e(i)=1/i: NEXT i
2990 LET term=0
3000 FOR j=2 TO ki
3010 LET psi=s(j-1): LET gamma=(hi+term)/psi
3020 LET eta=hi/psi: LET lim=kip1-j
3030 FOR i=1 TO lim: LET e(i)=gamma*e(i)-eta*e(i+1): NEXT i
3040 LET o(j)=e(1)
3050 LET term=psi: NEXT j
3060 DIM x(nrec)
3070 FOR j=1 TO ki
3080 LET i=kip1-j
3090 LET temp2=o(i)
3100 FOR l=1 TO nrec: LET x(l)=x(l)+temp2*h(l,i): NEXT l
3110 NEXT j
3120 FOR l=1 TO nrec: LET y(l)=y(l)+hi*x(l): NEXT l
3130 RETURN
```

13. Programare liniară

13.1 Simplex

Programul rezolvă probleme de programare liniară de forma

$$\text{minim} \sum_{j=1}^{n} c_j x_j$$

cu restricțiile

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad 1 \le i \le MI$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad MI < i \le m$$

$$x_j \ge 0, \quad 1 \le j \le n$$

Semnificațiile parametrilor sînt următoarele

m – număr total de restricții

n – număr de variabile

MI – număr de inegalități ($0 \le MI \le m$)

A(m,n) – matricea restricțiilor (cu componentele a_{ij}, $1 \le i \le m$, $1 \le j \le n$)

b(m) – termenul liber (vector de dimensiune m)

c(n) – coeficienții funcției obiectiv (vector de dimensiune n)

Observație. Introducerea datelor se realizează conversațional în timpul execuției programului.

•P66

```
 10 REM ****************************************************
 20 REM * Program BASIC pentru rezolvarea problemelor de   *
 30 REM * programare liniara cu algoritmul SIMPLEX .       *
 40 REM ****************************************************
 50 REM
 60 POKE USR "S", BIN 11111111
 70 POKE USR "S"+1, BIN 01100000
 80 POKE USR "S"+2, BIN 00110000
 90 POKE USR "S"+3, BIN 00011000
100 POKE USR "S"+4, BIN 00011000
110 POKE USR "S"+5, BIN 00110000
```

```
120 POKE USR "S"+6, BIN 01100000
130 POKE USR "S"+7, BIN 11111111
140 POKE USR "L", BIN 00000010
150 POKE USR "L"+1, BIN 00000100
160 POKE USR "L"+2, BIN 00001000
170 POKE USR "L"+3, BIN 00010000
180 POKE USR "L"+4, BIN 01001000
190 POKE USR "L"+5, BIN 00100100
200 POKE USR "L"+6, BIN 00010010
210 POKE USR "L"+7, BIN 00001000
220 POKE USR "G", BIN 01000000
230 POKE USR "G"+1, BIN 00100000
240 POKE USR "G"+2, BIN 00010000
250 POKE USR "G"+3, BIN 00001000
260 POKE USR "G"+4, BIN 00010010
270 POKE USR "G"+5, BIN 00100100
280 POKE USR "G"+6, BIN 01001000
290 POKE USR "G"+7, BIN 00010000
300 CLS : PRINT AT 2,0;"Programul  ** S I M P L E X ** "
310 PRINT "rezolva probleme de programare  liniara de forma:"
320 PRINT : PRINT : PRINT TAB 14;"n"
330 PRINT TAB 8;"minim S c(j)*x(j)": PRINT TAB 13;"j=1": PRINT
340 PRINT "  n": PRINT "  S A(i,j)*x(j) L b(i) , 1LiLMI"
350 PRINT " j=1": PRINT
360 PRINT "  n": PRINT "  S A(i,j)*x(j) = b(i) , MI<iLm"
370 PRINT " j=1"
380 PRINT : PRINT TAB 8;"x(j)G0 , 1LjLn": PAUSE 3000
390 CLS : PRINT AT 2,0;"Semnificatia parametrilor este"
400 PRINT "urmatoarea:": PRINT : PRINT
410 PRINT "m  -- numar total de restrictii"
420 PRINT "n  -- numar de variabile"
430 PRINT "MI -- numar de inegalitati"
440 PRINT TAB 8;"( 0 L MI L m )": PRINT
450 PRINT "A(m,n) - matricea restrictiilor"
460 PRINT "b(m) - termenul liber (vector)"
470 PRINT "c(n) - coeficientii functiei"
480 PRINT TAB 7;"obiectiv ": PRINT
490  PRINT AT 18,0;"Introduceti datele problemei:"
500 INPUT TAB 11;"m = ";m
510 INPUT TAB 11;"n = ";n
520 INPUT TAB 11;"MI = ";mi
530 DIM a(m,n): DIM b(m): DIM c(n)
540 FOR i=1 TO m: FOR j=1 TO n
550 INPUT ( TAB 8;"A(";i;",";j;") = ");a(i,j)
560 NEXT j: NEXT i
570 FOR i=1 TO m: INPUT ( TAB 8;"b(";i;") = ");b(i): NEXT i
580 FOR j=1 TO n: INPUT ( TAB 8;"c(";j;") = ");c(j): NEXT j
590 CLS : PRINT AT 3,3;" *** SIMPLEX: - START ***"
600 PRINT AT 8,10; FLASH 1;" REZOLVARE "
610 REM  S M P L X
620 LET inf=1E38: LET d=1E-8
630 DIM e(m): LET lk=0
640 FOR i=1 TO m
```

```
 650 IF b(i) >= 0 THEN GO TO 700
 660 LET b(i)=-b(i)
 670 FOR j=1 TO n
 680 LET a(i,j)=-a(i,j): NEXT j
 690 IF i <= mi THEN LET lk=lk+1: LET e(lk)=i
 700 NEXT i
 710 GO SUB 910: REM PLEXR
 720 CLS : IF mes>0 THEN GO TO 750
 730 PRINT   AT 10,0;"  Nu exista solutii admisibile!"
 740 PAUSE 100: GO TO 830
 750 IF mes=1 THEN GO TO 780
 760 PRINT AT 10,0;"Problema are o solutie infinita."
 770 PAUSE 150: GO TO 830
 780 PRINT "Valoarea minima a problemei este": PRINT
 790 PRINT TAB 8;"val. = ";val: PRINT
 800 PRINT "Solutia optima este:": PRINT
 810 FOR j=1 TO n: PRINT TAB 8;"x(";j;") = ";c(j)
 820 NEXT j: PAUSE 500
 830 PRINT : PRINT : PRINT "Doriti sa rezolvati o noua"
 840 PRINT "problema de programare"
 850 PRINT "liniara ? (y/n)"
 860 LET a$= INKEY$
 870 IF a$="y" THEN GO TO 390
 880 IF a$ <> "n" THEN GO TO 860
 890 CLS : PRINT AT 3,5;"*** SIMPLEX - STOP ***": PAUSE 200
 900 PRINT AT 10,10;"La revedere !": PAUSE 150: STOP
 910 REM   P L E X R
 920 DIM w(m,m): DIM i(m): DIM n(n+lk)
 930 DIM q(m): DIM p(m): DIM l(m): DIM z(m)
 940 FOR i=1 TO m
 950 LET w(i,i)=1: NEXT i
 960 LET val=0: LET imn=n+mi: LET mn=m+n
 970 FOR i=1 TO n: LET n(i)=i: NEXT i
 980 LET kota=n+lk
 990 IF mi=0 THEN GO TO 1060
1000 FOR i=1 TO mi: LET i(i)=n+i: NEXT i
1010 IF lk=0 THEN GO TO 1230
1020 FOR i=1 TO lk
1030 LET j=e(i): LET n(n+i)=n+j: LET i(j)=mn+j
1040 LET p(j)=1: LET val=val+b(j): NEXT i
1050 IF mi=m THEN GO TO 1100
1060 LET mi1=mi+1
1070 FOR i=mi1 TO m
1080 LET i(i)=n+i: LET p(i)=1
1090 LET val=val+b(i): NEXT i
1100 LET nf=1
1110 GO SUB 1450: REM TESTR
1120 IF cc <= 0 AND nf=1 THEN GO TO 1250
1130 IF cc <= 0 AND nf=2 THEN GO TO 1360
1140 GO SUB 1770: REM PIVOTR
1150 IF i0=0 AND nf=1 THEN GO TO 380
1160 IF i0=0 AND nf=2 THEN GO TO 490
```

```
1170 GO SUB 2210: REM TRANSR
1180 LET l=i(i0): LET i(i0)=j0
1190 IF l <= imn THEN GO TO 1220
1200 LET kota=kota-1
1210 FOR i=k0 TO kota: LET n(i)=n(i+1): NEXT i: GO TO 1110
1220 LET n(k0)=l: GO TO 1110
1230 IF mi=m THEN GO TO 1340
1240 GO TO 1060
1250 IF val=0 THEN GO TO 1270
1260 LET mes=0: RETURN
1270 FOR i=1 TO m
1280 IF i(i) <= n THEN LET val=val+c(i(i))*b(i)
1290 LET p(i)=0
1300 FOR j=1 TO m
1310 IF i(j) <= n THEN LET p(i)=  p(i)+c(i(j))*w(j,i)
1320 NEXT j
1330 NEXT i
1340 LET nf=2: GO TO 1110
1350 LET mes=1+j0: GO TO 1370
1360 LET mes=1
1370 FOR j=1 TO n
1380 FOR i=1 TO m
1390 IF j=i(i) THEN GO TO 1420
1400 NEXT i
1410 LET c(j)=0: GO TO 1430
1420 LET c(j)=b(i)
1430 NEXT j
1440 RETURN
1450 REM  T E S T R
1460 LET cc=0
1470 IF nf=2 THEN GO TO 1490
1480 LET eps=d: GO TO 1580
1490 LET eps=0: LET l=0
1500 FOR i=1 TO kota
1510 LET j=n(i)
1520 IF j>n THEN GO TO 1550
1530 IF c(j)=0 THEN GO TO 1550
1540 LET eps=eps+ ABS c(j): LET l=l+1
1550 NEXT i
1560 IF l=0 THEN GO TO 1480
1570 LET eps=eps/l*d
1580 FOR k=1 TO kota
1590 LET j=n(k)
1600 IF j <= n THEN GO TO 1670
1610 IF lk=0 THEN GO TO 1650
1620 FOR i=1 TO lk
1630 IF j-n=e(i) THEN GO TO 1660
1640 NEXT i
1650 LET r=p(j-n): GO TO 1710
1660 LET r=-p(j-n): GO TO 1710
1670 LET r=0
1680 FOR i=1 TO m
```

```
1690 IF p(i) <> 0 AND a(i,j) <> 0 THEN LET r=r+p(i)*a(i,j)
1700 NEXT i
1710 IF nf=2 AND j <= n THEN LET r=r-c(j)
1720 IF ABS r<eps THEN LET r=0
1730 IF r <= cc THEN GO TO 1750
1740 LET cc=r: LET j0=j: LET k0=k
1750 NEXT k
1760 RETURN
1770 REM  P I V O T R
1780 FOR i=1 TO m: LET l(i)=i: LET z(i)=b(i): NEXT i
1790 LET lm=m: LET nb=0
1800 IF j0 <= n THEN GO TO 1900
1810 LET j=j0-n
1820 IF lk=0 THEN GO TO 1860
1830 FOR i=1 TO lk
1840 IF j=e(i) THEN GO TO 1880
1850 NEXT i
1860 FOR i=1 TO m: LET q(i)=w(i,j): NEXT i
1870 GO TO 1980
1880 FOR i=1 TO m: LET q(i)=-w(i,j): NEXT i
1890 GO TO 1980
1900 FOR i=1 TO m
1910 LET q(i)=0
1920 FOR j=1 TO m
1930 IF w(i,j)=0 OR a(j,j0)=0 THEN GO TO 1970
1940 LET eps=w(i,j)*a(j,j0)
1950 LET q(i)=q(i)+eps
1960 IF ABS q(i)< ABS (eps*d) THEN LET q(i)=0
1970 NEXT j: NEXT i
1980 LET i0=0: LET min=inf
1990 FOR l=1 TO lm
2000 LET i=l(l)
2010 IF q(i) <= 0 THEN GO TO 2070
2020 LET rap=z(i)/q(i)
2030 IF rap <> min THEN GO TO 2050
2040 LET k=k+1: LET l(k)=i
2050 IF rap >= min THEN GO TO 2070
2060 LET min=rap: LET i0=i: LET k=1: LET l(1)=i
2070 NEXT l
2080 IF i0=0 THEN RETURN
2090 IF k=1 THEN RETURN
2100 LET lm=k
2110 LET nb=nb+1
2120 IF nb>kota THEN RETURN
2130 LET k=n(nb)
2140 IF k=j0 OR k>n THEN GO TO 2110
2150 FOR l=1 TO lm
2160 LET i=l(l): LET z(i)=0
2170 FOR j=1 TO m
2180 IF w(i,j) <> 0 OR a(j,k) <> 0 THEN LET z(i)=z(i)+w(i,j)*a(j,k)
2190 NEXT j: NEXT l
2200 GO TO 1980
```

```
2210 REM  T R A N S R
2220 FOR i=1 TO m
2230 IF q(i)=0 OR i=i0 THEN GO TO 2350
2240 LET piv=q(i)/q(i0)
2250 IF b(i0)=0 THEN GO TO 2290
2260 LET eps= ABS b(i)*d
2270 LET b(i)=b(i)-b(i0)*piv
2280 IF ABS b(i)<eps THEN LET b(i)=0
2290 FOR j=1 TO m
2300 IF w(i0,j)=0 THEN GO TO 2340
2310 LET eps= ABS w(i,j)*d
2320 LET w(i,j)=w(i,j)-w(i0,j)*piv
2330 IF ABS w(i,j)<eps THEN LET w(i,j)=0
2340 NEXT j
2350 NEXT i
2360 LET piv=cc/q(i0)
2370 IF b(i0)=0 THEN GO TO 2410
2380 LET eps= ABS val*d
2390 LET val=val-b(i0)*piv
2400 IF ABS val<eps THEN LET val=0
2410 FOR i=1 TO m
2420 IF w(i0,i)=0 THEN GO TO 2460
2430 LET eps= ABS p(i)*d
2440 LET p(i)=p(i)-w(i0,i)*piv
2450 IF ABS p(i)<eps THEN LET p(i)=0
2460 NEXT i
2470 LET piv=q(i0)
2480 IF b(i0) <> 0 THEN LET b(i0)=b(i0)/piv
2490 FOR i=1 TO m
2500 IF w(i0,i)=0 THEN GO TO 2520
2510 LET w(i0,i)=w(i0,i)/piv
2520 NEXT i
2530 RETURN
```

13.2 Transport

Programul rezolvă probleme de transport de forma

$$\text{minim} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

cu restricţiile

$$\sum_{j=1}^{n} x_{ij} = a_i \,, \qquad 1 \le i \le m$$

$$\sum_{i=1}^{m} x_{ij} = b_j \,, \qquad 1 \le j \le n$$

$x_{ij} \geq 0, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n$

Observaţie. Trebuie să fie îndeplinită condiţia

$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$ şi, de asemenea, $m \geq 2$, $n \geq 2$

Introducerea datelor se realizează conversaţional în timpul execuţiei programului.

• P67

```
 10 REM ***************************************************
 20 REM * Program BASIC pentru rezolvarea problemelor de *
 30 REM *                 T R A N S P O R T              *
 40 REM ***************************************************
 50 PRINT "Programul  TRANSPORT  rezolva"
 60 PRINT "probleme de transport de forma:"
 70 POKE USR "S", BIN 11111111
 80 POKE USR "S"+1, BIN 01100000
 90 POKE USR "S"+2, BIN 00110000
100 POKE USR "S"+3, BIN 00011000
110 POKE USR "S"+4, BIN 00011000
120 POKE USR "S"+5, BIN 00110000
130 POKE USR "S"+6, BIN 01100000
140 POKE USR "S"+7, BIN 11111111
150 PRINT : PRINT TAB 5;"minim S S C(i,j)*X(i,j)"
160 PRINT TAB 11;"i j": PRINT
170 PRINT "cu restrictiile": PRINT
180 PRINT TAB 5;"S X(i,j) = A(i) , 1<i<M": PRINT TAB 5;"j"
190 PRINT : PRINT TAB 5;"S X(i,j) = B(j) , 1<j<N"
200 PRINT TAB 5;"i": PRINT
210 PRINT FLASH 1;"ATENTIE !"; FLASH 0;
220 PRINT "  Trebuie sa fie"
230 PRINT "indeplinita conditia:"
240 PRINT : PRINT TAB 5;"S A(i) = S B(j)"
250 PRINT TAB 5;"i"; TAB 14;"j": PRINT "   1<i<M    1<j<N"
260 PRINT : PRINT "In plus: M >= 2 & N >= 2"
270 PAUSE 2000: CLS
280 PRINT "Introduceti datele problemei:"
290 INPUT "M=";M: PRINT "M = ";M
300 INPUT "N=";N: PRINT "N = ";N
310 LET MN=M+N-1
320 DIM A(M): DIM B(N): DIM C(M,N)
330 DIM X(MN): DIM I(MN): DIM J(MN)
340 PRINT "Vectorul ofertelor A(i), 1<i<M"
350 LET SA=0: FOR i=1 TO M
360 INPUT ("A(";i;")=");A(i): LET SA=SA+A(i): NEXT i
370 PRINT "Vectorul cererilor B(j), 1<j<N"
380 LET SB=0: FOR j=1 TO N
390 INPUT ("B(";j;")=");B(j): LET SB=SB+B(j): NEXT j
400 PRINT "Matricea coeficientilor de cost:"
410 PRINT "C(i,j), citita pe linii."
```

```
420 FOR i=1 TO M
430 PRINT "Linia ";i;" ..."
440 FOR j=1 TO N: INPUT ("C(";i;",";j;")=");C(i,j): NEXT j
450 NEXT i: CLS
460 PRINT AT 7,6; FLASH 1; BRIGHT 1;" R E Z O L V A R E "
470 GO SUB 620: CLS
480 PRINT "Solutia optima a problemei de   transport este:"
490 PRINT
500 LET S=0
510 FOR K=1 TO MN
520 LET S=S+X(K)*C(I(K),J(K))
530 PRINT TAB 8;"X(";I(K);",";J(K);") = ";X(K)
540 NEXT K
550 PRINT : PRINT "Costul optim de transport este:"
560 PRINT : PRINT TAB 8;"COST = ";S
570 PRINT AT 20,0;"Doriti rezolvarea altei"
580 PRINT "probleme ? (y/n)"
590 LET a$= INKEY$ : IF INKEY$ ="y" THEN CLS : GO TO 280
600 IF INKEY$ <> "n" THEN GO TO 590
610 CLS : PRINT AT 10,10; FLASH 1;" La revedere ! ": STOP
620 REM Subrutina TRPRT
630 LET INF=1E38: LET D=1E-8
640 LET NRBAZ=0
650 REM Determinarea unei solutii initiale de baza.
660 LET L=0: LET II=0: LET IJ=0
670 IF L=MN THEN GO TO 900
680 LET MIN=INF
690 FOR I=1 TO M
700 IF A(I)=INF THEN GO TO 760
710 FOR J=1 TO N
720 IF B(J)=INF THEN GO TO 750
730 IF MIN <= C(I,J) THEN GO TO 750
740 LET MIN=C(I,J): LET I0=I: LET J0=J
750 NEXT J
760 NEXT I
770 LET L=L+1: LET I(L)=I0: LET J(L)=J0: LET EPS=D
780 LET AMAX=A(I0): IF AMAX<B(J0) THEN LET AMAX=B(J0)
790 IF (A(I0) <> 0 OR B(J0) <> 0) THEN LET EPS=AMAX*D
800 LET R=A(I0)-B(J0)
810 IF ABS (R) <= EPS THEN LET R=0
820 IF R<0 THEN GO TO 880
830 IF R=0 THEN GO TO 850
840 IF R>0 THEN GO TO 860
850 IF M-II>N-IJ THEN GO TO 880
860 LET X(L)=B(J0): LET A(I0)=R: LET B(J0)=INF
870 LET IJ=IJ+1: GO TO 670
880 LET X(L)=A(I0): LET B(J0)=-R: LET A(I0)=INF
890 LET II=II+1: GO TO 670
900 REM Calculul multiplicatorilor simplex
910 PRINT AT 12,4;"S-a determinat o solutie"
920 PRINT TAB 7;"initiala de baza."
930 LET LS=0: LET LI=1
```

```
 940 LET IN=2: LET A(I(1))=0
 950 LET B(J(1))=C(I(1),J(1))
 960 LET O=LI
 970 LET IJ=I(O)
 980 FOR Q=IN TO MN
 990 IF IJ <> I(Q) THEN GO TO 1050
1000 LET L=Q
1010 GO SUB 1570
1020 LET B(J)=C(IJ,J)-A(IJ)
1030 LET IN=IN+1
1040 IF IN>MN THEN GO TO 1230
1050 NEXT Q
1060 LET O=O+1
1070 IF O <= LS THEN GO TO 970
1080 LET LI=LS+1: LET LS=IN-1
1090 LET Q=LI
1100 LET IJ=J(Q)
1110 FOR O=IN TO MN
1120 IF IJ <> J(O) THEN GO TO 1180
1130 LET L=O
1140 GO SUB 1570
1150 LET A(I)=C(I,IJ)-B(IJ)
1160 LET IN=IN+1
1170 IF IN>MN THEN GO TO 1230
1180 NEXT O
1190 LET Q=Q+1
1200 IF Q <= LS THEN GO TO 1100
1210 LET LI=LS+1: LET LS=IN-1
1220 GO TO 960
1230 REM Calculul costurilor comparative
1240 REM si testul de optimalitate
1250 LET MIN=INF
1260 FOR I=1 TO M
1270 FOR J=1 TO N
1280 LET AMAX= ABS C(I,J)
1290 IF AMAX< ABS A(I) THEN LET AMAX= ABS A(I)
1300 IF AMAX< ABS B(J) THEN LET AMAX= ABS B(J)
1310 LET EPS=AMAX*D
1320 LET R=C(I,J)-(A(I)+B(J))
1330 IF ABS R<EPS THEN LET R=0
1340 IF MIN <= R THEN GO TO 1360
1350 LET MIN=R: LET I0=I: LET J0=J
1360 NEXT J
1370 NEXT I
```

```
1380 IF MIN >= 0 THEN RETURN
1390 REM Determinarea unui ciclu
1400 LET LI=1: LET LS=1: LET J=J0
1410 FOR L=LS TO MN
1420 IF J=J(L) THEN GO TO 1460
1430 NEXT L
1440 LET IN=LI: LET L=LS-1: GO SUB 1570
1450 LET LI=LI+1: LET I=I(LS-1): GO TO 1500
1460 LET IN=LS: GO SUB 1570
1470 LET I=I(LS)
1480 IF I=I0 THEN GO TO 1650
1490 LET LS=LS+1
1500 FOR L=LS TO MN
1510 IF I=I(L) THEN GO TO 1550
1520 NEXT L
1530 LET IN=LI: LET L=LS-1: GO SUB 1570
1540 LET LI=LI+1: LET J=J(LS-1): GO TO 1410
1550 LET IN=LS: GO SUB 1570
1560 LET J=J(LS): LET LS=LS+1: GO TO 1410
1570 LET I=I(L): LET J=J(L)
1580 IF IN >= L THEN RETURN
1590 LET R=X(L)
1600 LET K=L-1: LET X(L)=X(K)
1610 LET I(L)=I(K): LET J(L)=J(K): LET L=K
1620 IF L>IN THEN GO TO 1600
1630 LET X(IN)=R: LET I(IN)=I: LET J(IN)=J
1640 RETURN
1650 REM Schimbarea bazei
1660 LET NRBAZ=NRBAZ+1
1670 PRINT AT 16,5;"Schimbarea bazei nr.";NRBAZ
1680 LET MIN=INF
1690 FOR L=LI TO LS STEP 2
1700 IF MIN <= X(L) THEN GO TO 1720
1710 LET MIN=X(L): LET I=L
1720 NEXT L
1730 LET I(I)=I0: LET J(I)=J0: LET EPS=D*MIN
1740 FOR L=LI TO LS STEP 2
1750 LET X(L)=X(L)-MIN
1760 IF X(L)<EPS THEN LET X(L)=0
1770 NEXT L
1780 LET X(I)=MIN: LET LI=LI+1
1790 FOR L=LI TO LS STEP 2
1800 LET X(L)=X(L)+MIN
1810 NEXT L
1820 GO TO 900
```

14. Metode de sortare

În cadrul multor aplicatii,apare frecvent ca subproblemă, sortarea crescătoare sau descrescătoare a unei secvenţe de numere sau, mai general, sortarea unei secvenţe de obiecte aparţinînd unui domeniu pe care este definită o relaţie de ordine totală.

Pentru simplificare, vom considera numai problema sortării crescătoare a unei secvenţe de numere. Evident, lejere modificări permit adaptarea unui astfel de algoritm la cazul general.

Exista o clasă bogată de algoritmi care permit rezolvarea acestei probleme. Vom considera în cele ce urmează numai trei dintre metodele foarte cunoscute şi anume: sortarea cu bule sau "bubble sort", sortarea rapidă sau "quick sort" şi sortarea binară.

14.1 Bubble Sort

Metoda de sortare se bazează pe o idee foarte simplă şi anume: componentele consecutive din secvenţa de sortat sînt comparate succesiv, fiecărei perechi de numere comparate şi care nu verifică condiţia ca prima componentă să fie mai mică sau cel mult egală cu cea de a doua fiindu-i aplicată operaţia de interschimbare a valorilor corespunzatoare acestora.

Evident, o singură "traversare" a secvenţei de sortat nu garantează obţinerea unei secvenţe sortate crescător, deci eventual este necesară o reluare a acestui calcul.

Secvenţa curentă îndeplineşte condiţia de a fi sortată crescător, dacă în urma efectuării unei traversări toate perechile de componente consecutive îndeplinesc condiţia ca prima componentă să fie mai mică sau cel mult egală cu cea de a doua.

Programul următor permite rezolvarea problemei sortării ascendente a unei secvenţe de numere dată pe baza algoritmului descris.

•P68

```
10 REM *****************************************************
20 REM * Program pentru sortarea unei secvente de numere  *
30 REM * in sens crescator. Datele de intrare:            *
40 REM *  - Numarul componentelor secventei - variabila N *
50 REM *  - Secventa de sotat - vectorul X                *
60 REM *****************************************************
70 INPUT "Numarul componentelor secventei:";n
80 DIM x(n)
```

```
 90 PRINT "Introduceti componentele secventei de sortat !"
100 FOR i=1 TO n
110 INPUT ("Componenta ";i;" = ");x(i)
120 NEXT i
130 PRINT : PRINT "Alegeti metoda de sortare:": PRINT
140 PRINT "pentru Buble sort ---> tastati "; FLASH 1;"B"
150 PRINT "pentru Hoare      ---> tastati "; FLASH 1;"H"
160 IF INKEY$ ="B" THEN GO SUB 240: GO TO 190
170 IF INKEY$ ="H" THEN GO SUB 360: GO TO 190
180 GO TO 160
190 PRINT : PRINT : PRINT "Secventa sortata este :": PRINT
200 FOR i=1 TO n
210 PRINT x(i);"  ";
220 NEXT i
230 STOP
240 REM ******************************************
250 REM * Subprogram pentru metoda "buble sort"  *
260 REM ******************************************
270 PRINT : PRINT TAB 10; FLASH 1;"buble sort": PRINT
280 LET sw=0
290 FOR i=1 TO n-1
300 IF x(i) <= x(i+1) THEN GO TO 330
310 LET sw=1
320 LET b=x(i): LET x(i)=x(i+1): LET x(i+1)=b
330 NEXT i
340 IF sw <> 0 THEN GO TO 280
350 RETURN
```

14.2 QUICK SORT

Una dintre metodele cele mai eficiente de sortare a fost imaginată de către Hoare, algoritmul de sortare bazat pe această idee fiind referit ca algoritmul de **sortare rapidă** sau "**quick sort**".

Fie L lista numerelor de sortat; x prima componenta a listei L;

dacă lista L conţine numai pe x, atunci L este sortată; altfel, fie L' lista rezultată din L prin eliminarea componentei x.

Metoda constă în obţinerea listelor L1,L2 astfel încît L1 conţine toate componentele listei L' mai mici sau egale decît x, L2 conţine toate componentele din L' mai mari decît x. Dacă L1' este lista L1 sortată crescător şi L2' este lista L2 sortată de asemenea crescător, atunci lista obţinută prin concatenarea (alipirea) listelor L1', lista constînd numai din x şi lista L2' reprezintă lista L sortată crescător. Sortarea listelor L1' şi L2' este realizată prin aplicarea aceluiaşi procedeu.

•P69

```
360 REM ***********************************************
370 REM * Subprogram pentru sortarea unei secvente de   *
380 REM * numere pe baza metodei Hoare.                 *
390 REM * Datele de intrare :                           *
400 REM *  - Secventa de sortat - vectorul X            *
410 REM *  - Lungimea secventei - variabila N           *
420 REM * Variabile si vectori de lucru :               *
430 REM *  - Vectorul L contine pozitiile in X a primei *
440 REM *    si ultimei componente din L1               *
450 REM *  - Vectorul R contine pozitiile in X a primei *
460 REM *    si ultimei componente din L2               *
470 REM *  - Variabilele LS, RS reprezinta pozitiile    *
480 REM *    extremitatilor listei curente prelucrate   *
490 REM *  - Pivotul PV de efectuare a comparatiilor    *
500 REM *  - PT indica pozitia vectorilor L, R asociata *
510 REM *    ultimei separari                           *
520 REM ***********************************************
530 PRINT : PRINT TAB 12; FLASH 1;"Hoare": PRINT
540 DIM l(n): DIM r(n)
550 LET l(1)=1: LET r(1)=n
560 REM Pozitiile in L, R corespunzatoare ultimei
570 REM liste sortate
580 LET ptz=1
590 REM Extremitatea ltinga a listei
600 LET lz=l(ptz)
610 REM Salvare valoare extremitate stinga
620 LET lsz=lz
630 REM Extremitatea dreapta a listei
640 LET rz=r(ptz)
650 REM Salvare valoare extremitate dreapta
660 LET rsz=rz
670 LET ptz=ptz-1
680 REM Daca lista curenta este compusa dintr-un singur
690 REM element, atunci lista este sortata
700 IF rz-lz <= 0 THEN GO TO 1300
710 REM Pivotul este plasat temporar pe extremitate stinga
720 LET pv=x(lz)
730 REM Daca lista are mai mult de 5 elemente,
740 REM pivotul este plast pe componenta mediana
750 IF rz-lz<5 THEN GO TO 870
760 LET med= INT ((lz+rz)/2)
770 IF pv<x(rz) AND pv>x(med) THEN GO TO 870
780 IF pv<x(med) AND pv>x(rz) THEN GO TO 870
790 IF x(rz)<pv AND x(rz)>x(med) THEN GO TO 850
800 IF x(rz)<x(med) AND x(rz)>pv THEN GO TO 850
810 REM Pivotul este plasat pe componenta mediana
820 LET pv=x(med)
830 LET x(med)=x(lz)
840 GO TO 870
850 LET pv=x(rz)
860 LET x(rz)=x(lz)
870 IF lz=rz THEN GO TO 1010
```

```
 880 IF pv>x(rz) THEN GO TO 950
 890 LET rz=rz-1
 900 GO TO 870
 910 IF lz >= rz THEN GO TO 1010
 920 IF pv<x(lz) THEN GO TO 980
 930 LET lz=lz+1
 940 GO TO 910
 950 LET x(lz)=x(rz)
 960 LET lz=lz+1
 970 GO TO 910
 980 LET x(rz)=x(lz)
 990 LET rz=rz-1
1000 GO TO 870
1010 LET x(lz)=pv
1020 IF lz-lsz <= rsz-lz THEN GO TO 1160
1030 IF rsz-lz <= 1 THEN GO TO 1110
1040 LET l(ptz+1)=lsz
1050 LET r(ptz+1)=lz-1
1060 LET lz=lz+1
1070 LET lsz=lz
1080 LET rz=rsz
1090 LET ptz=ptz+1
1100 GO TO 700
1110 IF lz-lsz <= 1 THEN GO TO 1300
1120 LET rz=lz-1
1130 LET rsz=lz
1140 LET lz=lsz
1150 GO TO 700
1160 IF lz-lsz <= 1 THEN GO TO 1240
1170 LET l(ptz+1)=lz+1
1180 LET r(ptz+1)=rsz
1190 LET rz=lz-1
1200 LET rsz=rz
1210 LET lz=lsz
1220 LET ptz=ptz+1
1230 GO TO 700
1240 IF rsz-lz <= 1 THEN GO TO 1300
1250 LET lz=lz+1
1260 LET lsz=lz
1270 LET rz=rsz
1280 GO TO 700
1290 REM Test daca exista inca liste de prelucrat
1300 IF ptz >= 1 THEN GO TO 600
1310 RETURN
```

14.3 Sortare binară

Programul realizează sortarea crescătoare a unei secvenţe de numere. Metoda utilizată constă în generarea unui arbore binar cu proprietăţile următoare:

1. informaţia asociată fiecărui vìrf este unul din numerele secvenţei.

2. pentru fiecare vîrf care are descendenți la stînga și/sau la drapta avem:

s<v≤d, unde:

v=informația asociată vîfului;

s=informația asociată descendentului la stînga;

d=informația asociată descendentului la dreapta.

Sortarea secvenței constă în efectuarea unei traversări a arborelui generat conform strategiei: subarbore stîng - vizitează rădăcina - subarbore drept.

•P70

```
  5 REM SORTARE BINARA
 10 INPUT "INDICATI NUMARUL DE COMPONENTE DIN
    SECVENTA DE SORTAT ";N
 20 DIM A(N): DIM T(N,4)
 30 PRINT FLASH 1; AT 10,10; "INTRODUCETI NUMERELE":
    PRINT AT 15,10;"ASTEPTATI PUTIN !":PAUSE 200:CLS
 40 FOR I=1 TO N:
    INPUT ("AL ";I;"-LEA NUMAR = ");A(I): NEXT I
 50 LET K=1: LET T(1,1)=A(1): LET T(1,4)=1: LET A$="¤¤¤"
 60 FOR I=2 TO N
 70 FOR J=1 TO K: IF A(I)=T(J,1) THEN
    LET T(J,4)=T(J,4)+1:GO TO 90
 80 NEXT J
 85 LET K=k+1: LET T(K,1)=A(I): LET T(K,4)=1
 90 NEXT I
100 FOR I=2 TO K
110 LET A=1
115 IF T(I,1)<T(A,1) THEN GO TO 130
120 IF T(A,3)=0 THEN LET T (A,3)=I: GO TO 200
125 LET A=T(A,3): GO TO 115
130 IF T (A,2)=0 THEN LET T(A,2)I: GO TO 200
140 LET A=T(A,2): GO TO 115
200 NEXT I
210 PRINT AT 0,3; FLASH 1;"TABELA DE REPREZENTARE A
    ARBORELUI": PRINT : PRINT
215 FOR I=1 TO K: PRINT INK 0;T(I,1); A$;
    INK 2;T(I,2);A$; INK 4;T(I,3): NEXT I
216 PRINT AT 20,1; "APASATI ORICE TASTA DACA DORITI
    CONTINUAREA": PAUSE 0
220 CLS
225 DIM S (1000,2): LET NS=0: DIM V(2,K): LET NV=0
230 LET A=1
231 IF T(A,2)<> O THEN GO TO 270
235 IF T(A,3)<> O THEN LET NS=NS+1: LET S (NS,1)=A:
    LET S (NS,2)=2: LET A=T (A,3): LET NV=NV+1:
    LET V(1,NV)=T(S(NS,1),1):
    LET V(2,NV)=T(S(NS,1),4): GO TO 231
```

```
240 LET NV=NV+1: LET V(1,NV)=T(A,1): LET V(2,NV)=T(A,4)
245 IF NS=O THEN GO TO 300
250 IF S (NS,2)=2 THEN LET NS=NS-1: GO TO 245
255 LET S(NS,2)=2: LET NV=NV+1:
    LET V(1,NV)=T(S(NS,1),1):
    LET V(2,NV)=T(S(NS,1),4)
260 IF T(S(NS,1),3)<>0 THEN LET A=T(S(NS,1),3):
    GO TO 231
265 LET NS=NS-1: GO TO 245
270 LET NS=NS+1: LET S(NS,1)=A: LET S(NS,2)=1: LET
    A=T(A,2): GO TO 231
300 PRINT AT 2,2; FLASH 1;"SECVENTA ORDONATA": PRINT:
    PRINT
310 FOR I=1 TO NV: FOR J=1 TO V(2,I): PRINT V(1,I); A$;:
    NEXT J: NEXT I
320 PRINT AT 20,0; FLASH 1; "APASATI ORICE TASTA DACA
    DORITI CONTINUAREA": PAUSE O: CLS
330 INPUT "DORITI RELUAREA PROGRAMULUI ? (DA/NU) ";B$:
    IF B$="DA" THEN CLS: GO TO 10
340 FOR I=1 TO 10: PRINT AT 10,10; FLASH 1;
    "LA REVEDERE": BEEP RND,I ^ 2*RND-20:
    NEXT I: CLS: STOP
```

15. Jocuri

15.1. Aveți spirit de observație ?

Programul care urmează, poate fi utilizat ca un mic test pentru verificarea spiritului de observație. Pe ecran vor fi afisate 10 litere; aveți la dispoziție cîteva momente pentru a încerca să le memorați. In continuare, pe ecran vor apare numai 9 din cele 10 litere afișate inițial. Puteți preciza care din litere a fost omisă ? Încercați.

●P71

```
  5 REM ************************************************
 10 REM *       Aveti spirit de observatie ?           *
 15 REM ************************************************
 20 DIM a(20)
 30 DIM b(20)
 40 FOR r=1 TO 20
 50 LET a(r)=r+37
 60 NEXT r
 70 FOR r=1 TO 10
 80 LET b(r)=a(INT(RND*20)+1)
 90 NEXT r
100 CLS
110 FOR r=1 TO 10
120 PRINT AT r,r*2;CHR$ b(r)
130 NEXT r
140 LET w=INT(RND*10)+1
150 LET m=b(w)
160 LET b(w)=0
170 FOR r=1 TO 126
180 PLOT r/2,1
190 NEXT r
200 CLS
210 FOR r=1 TO 10
220 PRINT AT r,r*2;CHR$ b(r)
230 NEXT r
240 PRINT " Care litera lipseste ? "
250 INPUT c$
260 IF CODE(c$)=m THEN GOTO 330
270 PRINT
280 PRINT
290 PRINT " *** Ati gresit !*** - Litera care lipseste"
300 PRINT
310 PRINT "                          este-"; CHR$ m;"-"
320 GOTO 350
```

```
330 PRINT
340 PRINT "******* CORECT !*******"
350 PRINT
360 PRINT "Doriti sa mai jucati ?(d/n)"
370 INPUT a$
380 IF a$="d" THEN RUN
390 STOP
```

15.2 Papagalul

"Papagalul" este un joc simplu şi amuzant prin care puteţi să verificaţi cît de rapid puteţi reacţiona. Odată competiţia începută, calculatorul va afişa o literă, iar dumneavoastră va trebui să tastaţi aceeaşi literă cît mai repede cu putinţă. După aproximativ 10 secunde, calculatorul va afişa scorul pe care l-aţi obţinut.

•P72

```
  5 REM ***********************************************
  6 REM *         PAPAGALUL                           *
  7 REM ***********************************************
 10 CLS :RESTORE:PLOT 100,90
 20 FOR i=1 TO 7 :READ x:READ y:READ a:DRAW x,y,a
    :NEXT i
 30 PLOT 89,139: DRAW 15,-10,-PI
 40 FOR i=0 TO 5 STEP 0.2 :CIRCLE 95,131,i :NEXT i
100 DEF FN t()=INT((65536*PEEK 23674 +256*PEEK 23673.+
     PEEK 23672)/50)
110 PRINT AT 14,0;" Apasati orice tasta cind vreti sa
     incepem !": IF INKEY$="" THEN GOTO 110
120 LET a=97: LET b=25:LET t1>=FN t(): LET c=0
130 LET x$=CHR$ (a+b*RND): LET a$=x$: PAUSE 1
140 PRINT AT 16,15;x$;
150 LET t2=FN t() : IF t2-t1>=10 THEN GOTO 200
160 LET x$=" ": PAUSE 10: IF INKEY$ <> a$ THEN GOTO 140
170 LET c=c+1: GOTO 130
200 PRINT AT 18,5;"Scorul dumneavoastra este  ";c
210 LET t=INT(t2-t1): PRINT AT 19,5;"in ";t;" secunde"
220 INPUT " Doriti sa mai jucati ?(d/n)";a$
230 IF a$="d" THEN GOTO 10
240 STOP
300 DATA -5, 8, 0.5, -8, -2, 1, 5, 7, -1, -10, -15, 2,
     10, 35,  -3, 40, 5, -4, -15, -40, 0.5
```

15.3 Simularea deplasarii unui vehicol

Stimularea deplasării controlate (cu tastele Z şi M) a unui vehicol pe o traiectorie marcată de asteri[redacted]uri este realizată pe baza utilizării instrucţiunii POKE 23692, -1 şi a funcţiei INKEY$

•P73

```
  1 PRINT
  5 REM SIMULEAZA MERSUL UNEI MASINI PE STRADA
 11 RESTORE: PRINT FLASH 1; AT 10,1;"TASTELE DE CONTROL
    SINT (Z,M)"
 12 PRINT FLASH 1; AT 15,2;"APASATI ORICE TASTA DACA
     DORITI CONTINUAREA": PAUSE 0: CLS
 20 LET C=0
 30 LET T=0
 40 GO SUB 250
 50 LET A=10
 60 LET X=13
 70 LET Y=12
 80 LET K=INT (RND*2)
 90 LET A=A-(K=1 AND A>1)+(K=0 AND A<24)
100 REM URMATOAREA LINIE DE PROGRAM CONTINE LITERA"C"
     IN MODUL GRAFIC. ACELASI "C" SE AFLA SI IN LINIA 200
110 PRINT AT Y,X-1; INK 1;"A"
120 PRINT AT 20,A; INK 2;"*";TAB A+5;"*"
130 PRINT
140 POKE 23692,-1: PRINT
150 PRINT INK 6; PAPER 2; AT 0,10;"SCORUL ESTE ";T;" "
160 IF SCREEN$ (Y+1,X-1)="*" THEN GO TO 200
170 LET X=X-(INKEY$="Z")+(INKEY$="M")
180 LET T=T+1
190 GO TO 80
200 PRINT AT Y,X-1;INK 1;"A"
210 FOR W=1 TO 50: PRINT AT 6,8; FLASH 1; BRIGHT 1;
     "ESTI IN DECOR !!!"
220 PRINT AT 8,10; FLASH 1; BRIGHT 1; INK RND*7; PAPER
     9;" SCORUL TAU ESTE ";T
230 BEEP .01, RND*20-RND*20
235 NEXT W
240 CLS: INPUT "DORITI RELUAREA ? ";A$: IF A$="DA" THEN
     GO TO 11
245 CLS: PRINT AT 10,10; FLASH 1; "LA REVEDERE": PAUSE
     300: CLS: STOP
250 FOR J=0 TO 7
260 READ Z
270 POKE USR "A"+J,Z
289 NEXT J
```

```
290 RETURN
300 DATA BIN 00110110, BIN 00110110, BIN 00110110,
    BIN 00010100, BIN 00111110, BIN 00110110,
    BIN 00011100,0
```

15.4 GROTA DRAGONULUI

Programul realizează o mişcare aleatoare a unei figuri ("dragon") într-un pătrat format din 10x10 puncte. După îndeplinirea unei anumite condiţii, devine liber şi programul se opreşte.

•P74

```
 10 REM GROTA DRAGONULUI
 20 DIM A (10,10): LET M=0
 30 GO SUB 500
 40 LET X=INT (RND*2)
 50 IF X=0 THEN LET P=P+1
 60 IF X=1 THEN LET P=P-1
 70 LET X=INT (RND*2)
 80 IF X=0 THEN LET Q=Q+1
 90 IF X=1 THEN LET Q=Q-1
100 IF Q<1 THEN LET Q=Q+1
110 IF P<1 THEN LET P=P+1
120 LET M=M+1
130 PRINT AT 3,3; INVERSE 1; "INCERCAREA ";M; " "
410 LET A(P,Q)=98
415 PRINT AT 5,5;
420 FOR X=1 TO 10
430 FOR Y=1 TO 10
440 PRINT FLASH 1*(A(X,Y)=98); INK P/2;
    CHR$ (A(X,Y)+46);
450 NEXT Y
460 PRINT: PRINT TAB 5;
470 BEEP 0.008,x*y/2: NEXT x
480 LET a(p,q)=0
490 IF q>8 OR p>8 THEN GO TO 600
495 GO TO 40
500 LET q=INT (RND*3)+4: LET p=INT (RND*3)+4
510 RANDOMIZE
520 FOR b=0 TO 7
530 READ c
540 POKE USR "A"+b,c
550 NEXT b
560 BRIGHT 1: PAPER 7: INK 0: CLS
570 BORDER 2
580 RETURN
590 DATA 35, 70, 76,95, 255, 223, 18, 51
```

```
600 PRINT BRIGHT 1; FLASH 1; INK 0; PAPER 4; TAB 5;
    "IN SFIRSIT LIBER ..."
```

15.5 CURSA

Este un joc simplu în care jucătorul este solicitat să ghicească numărul care va ieşi cîştigător. Jocul se desfăşoară între om şi calculator, calculatorului fiindu-i indicat unul din numerele 1, 2, 3, ..., 9.

•P75

```
  1 REM "CURSA"
  3 LET K=0: LET P=0
  5 RANDOMIZE
  7 GO SUB 200
 10 FOR X=1 TO 22
 20 PRINT INK 4; TAB 30; "▞"
 30 NEXT X
 35 PRINT AT 0,6;"ATI PARIAT CA VA CISTIGA NUMARUL ";W
 40 DIM A (9)
 50 FOR X=1 TO 9
 60 PRINT AT 2*X,A(X);" "
 70 LET A(X)=A(X)+RND*2
 80 PRINT AT 2*X,A(X); INK X/2;X
 85 BEEP .01,3*X
 90 IF A(X)>30 THEN GO TO 115
100 NEXT X
110 GO TO 50
115 FOR G=1 TO 50 STEP 2
120 PRINT AT 18,6; INK RND*7; "NUMARUL ";X;
    " A CISTIGAT CURSA!"
123 BEEP .02,G
125 IF W=X THEN PRINT AT 20,3; INK RND*7; "SI
    DUMNEAVOASTRA ATI CISTIGAT !! "
130 FLASH RND
140 NEXT G
142 IF W=X THEN LET K=K+1
145 FLASH 0
150 INPUT "DORITI RELUAREA JOCULUI ? (DA/NU)";A$:
    IF A$= DA" THEN CLS: GO TO 5
152 CLS: PRINT AT 10,10; "ATI CISTIGAT DE ";K;
    " ORI DIN ";P;"ETAPE"; FLASH 1; INVERSE 1;
    AT 15,10; "LA REVEDERE !": BEEP RND*5,RND*10
153 CLS : STOP
200 BORDER 0
205 PRINT AT 3,1; INK 2; "BINE ATI VENIT LA CONCURS!"
210 PRINT AT 5,6; INK 4;"IN CURSA PARTICIPA 9
    CONCURENTI"
```

```
220 INPUT "CE NUMAR PARIATI CA VA CISTIGA ?";W
230 BORDER 2
233 LET P=P+1
240 CLS: RETURN
```

15.6 COMPETIŢIA ÎNTRE DOUĂ SUBMARINE

Jocul constă în stimularea unei întreceri între două submarine dirijate aleator de către calculator.

Submarinul din partea superioară a ecranului este al calcularorului cel din partea inferioară este al dumneavoastră.

Cîştigă submarinul care atinge primul marginea ecranului.

• P76

```
10 REM PROGRAMUL SIMULEAZA INTRECEREA INTRE DOUA
   SUBMARINE
11 REM CELE DOUA SUBMARINE SINT DIRIJATE ALEATOR
   DE CATRE CALCULATOR
12 REM PRIMUL SUBMARIN ESTE AL CALCULTORULUI, CEL DE
   AL DOILEA ESTE AL DUMNEAVOASTRA
15 PAPER 5: BORDER 5: CLS
20 LET a$ = "
```



```
                                  "
30 LET c=28
40 LET h=28
50 LET x=5
55 BEEP .01,RND*50
60 GO SUB 100
70 LET x=10
75 BEEP .01,RND*50
80 GO SUB 100
90 GO TO 50
100 IF x=5 THEN LET c=c-RND: PRINT AT x,c; INK 6; a$:
    IF c<2 THEN PRINT AT 0,0; PAPER 6; BRIGHT 1;
    "CALCULATORUL A CISTIGAT": STOP
110 IF x=10 THEN LET h=h-RND: PRINT AT x,h; INK 2; a$:
    IF h<2 THEN PRINT AT 0,0; PAPER 6; FLASH 1;
    "DUMNEAVOASTRA ATI CISTIGAT": STOP
120 RETURN
```

15.7 INVADATORII

Jocul este o versiune simplificată a unui joc foarte cunoscut pentru ilustrarea unor modalități de utilizare a caracterelor definite de utilizator. Evitarea "invadatorilor" care descind din partea superioară a ecranului pentru a vă ataca, poate fi realizată prin acționarea tastelor 5 și 8. Puteți de asemenea ataca invadatorii acționînd tasta 7. Dacă sînteți un bun țintaș, atunci invadatorii vor dispare de pe ecran.

●P77

```
  1 REM INVADATORII
  4 BORDER 2: PAPER 0
  5 GO SUB 290
 10 DIM A$ (32)
 20 DIM B$ (32)
 30 FOR D=1 TO 7
 40 LET A$=" F F F F F "
 50 LET X=INT (RND*32)
 60 LET C=X
 70 PRINT AT 5,12; INK X/5; PAPER 9; FLASH 1;
   BRIGHT 1; "VALUL ";D;"  "
 80 LET C=0
 90 FOR B=D+9 TO 19 STEP 2
100 FOR A=0 TO 31
110 LET X=X+(INKEY$="9" AND X<31)-
    (INKEY$="8" AND X>0)
120 PRINT AT B,0; INK RND*2;A$; 20,C;" " AND C<>X;
    AT 20,X; INK 4; "B"
130 IF A$=B$ THEN GO TO 200 +(60 AND D=7)
140 LET C=X
150 IF INKEY$="7" THEN BEEP .01,20:
    IF A$ (X+1)="F" THEN LET A$ (X+1)="█":
    PRINT AT B,0; INK 6; A$: BEEP .01,50:
    LET AS(X+1)=" ": LET SC=SC+1:
    PRINT AT 2,10; INK 2;
    PAPER 0; BRIGHT 1; FLASH 1; "SCORUL"; SC*27187
160 NEXT A
165 LET A$=A$ (4 TO)+A$ ( TO 3)
170 PRINT AT B,0; B$
180 NEXT B
190 GO TO 240
200 FOR B=1 TO 10
210 NEXT B
220 PRINT AT 20,C; " "
230 NEXT D
240 PRINT FLASH 1; INK RND*7; PAPER 9;"AU ATERIZAT"
250 BEEP .1, -RND*30: POKE 23692,-1: GO TO 240
```

```
260 PRINT INK RND *7; PAPER 9; "INVADATORII AU FOST
    DISTRUSI"
265 POKE 23692, -1
270 BEEP .01, RND*50: GO TO 260
290 FOR J=0 TO 7
300 READ Q: POKE USR "F"+J,Q
320 NEXT J
330 DATA BIN 00111110, BIN 00101010, BIN 00111110,
    BIN 00011100, BIN 00001000, BIN 01110111,
    BIN 01000001,BIN 01000001
350 FOR J=0 TO 7
360 READ Q: POKE USR "B"+J,Q
370 NEXT J
380 DATA BIN 01111110, BIN 01111110, BIN 01111110,
    BIN 01111110, BIN 01100110, BIN 11100011,
    BIN 11000011, BIN 10000001, BIN 10000001
400 LET SC=0
410 RETURN
```

15.8 Amplasarea reginelor pe tabla de şah

Programul determină combinaţiile posibile de aşezare a reginelor pe o tablă de şah de dimensiune dată n x n, astfel ca acestea să nu se atace reciproc.

•P78

```
 10 REM REZOLVA PROBLEMA DETERMINARII
    CONFIGURATIILOR POSIBILE PENTRU ASEZAREA
    A N REGINE PE O TABLA DE SAH NXN
 30 DIM A(40)
 35 LET D=0
 40 INPUT "INDICATI NUMARUL N (NUMARUL DE REGINE)",AO
 50 IF AO<4 OR AO>40 OR AO<>INT (AO) THEN GO TO 40
 60 LET Z=1: PRINT : PRINT
 70 IF A(Z)=AO THEN GO TO 180
 80 LET A(Z)=A(Z)+1
 90 IF Z=1 THEN GO TO 150
100 LET I=0
110 LET I=I+1
120 IF A (I)=A(Z) THEN GO TO 70
130 IF ABS (A(I)-A(Z))=ABS(I-Z) THEN GO TO 70
140 IF I<Z-1 THEN GO TO 110
150 IF Z=AO THEN GO TO 220
160 LET Z=Z+1
170 GO TO 70
180 LET A(Z)=0
190 LET Z=Z-1
200 IF Z=0 THEN CLS: PRINT AT 10,10;"TERMINAT":
```

```
     FOR W=1 TO 10: BEEP 5*RND,-20*RND:NEXT W:
     CLS: STOP
 210 GO TO 70
 220 LET D=D+1
 225 FOR W=1 TO 10: BEEP .3*RND,W*RND: NEXT W
 230 PRINT FLASH 1; AT 10,5;"COMBINATIA NUMARUL";
     FLASH 0;D
 235 PRINT AT 19,0; FLASH 1; "APASATI ORICE TASTA DACA
     DORITI AFISAREA URMATOAREI CONFIGURATII":
     PAUSE 0: CLS
 240 PRINT
 250 FOR M=1 TO AO
 260 FOR N=1 TO AO
 270 IF A(N)=M THEN PRINT "D";: GO TO 290
 280 PRINT "+ ";
 290 NEXT N
 300 PRINT
 310 NEXT M
 320 PRINT
 330 PRINT
 340 GO TO 70
```

15.9 HASAMI SHOGI

Este o variantă a unui joc oriental foarte vechi care se înrudește cu jocul de dame. Jocul se desfășoară între om și calculator prin efectuarea unor mutări succesive, prin indicarea coordonatelor careurilor sursă și respectiv destinație.

•P79

```
 10 REM HASAMI SHOGI
 20 GO SUB 630
 30 GO SUB 80
 40 GO SUB 360
 50 GO SUB 510
 60 GO SUB 360
 70 GO TO 30
 80 PRINT AT 0,0; "EU ALEG UN PATRAT"
 90 FOR A=99 TO 11 STEP -1: LET Q=INT (A/10):
    PRINT AT 0,24; CHR$ (64+Q); A-10*Q
100 IF A(A)<> C THEN GO TO 140
112 IF A(A-10)=E THEN IF A(A-9)=H THEN IF A(A-8)=C
    THEN BEEP 1,50: LET B=A-10: GO TO 280
115 IF A(A-10)=E THEN IF A(A-11)=H THEN
    IF A(A-12)=C THEN BEEP 1,50: LET B=A-10: GO TO 280
116 IF A (A-10)=E THEN IF A(A+11)=H THEN IF A(A+12)=C
    THEN BEEP 1,50: LET B=A-10: GO TO 280
```

```
117 FOR B=1 TO 4: IF A+2*C(B)<11 OR A+2*C(B)>99 THEN
    GO TO 130
120 IF A(A+C(B))=E AND A(A+2*C(B))=H AND A(A+3*C(B))=C
   THEN GO TO 260
130 NEXT B
140 NEXT A: PRINT AT 0,0;"
160 LET CO=0
170 LET CO=CO+1: BEEP .007,CO/4
180 LET A=INT (RND*89)+11
190 IF A(A)=C THEN BEEP .05.1: GO TO 220
200 IF CO<200 THEN GO TO 170
210 PRINT AT 0,0;"MAESTRUL HASAMI SHOGI VA DECLARA
   INVINGATOR": STOP
220 FOR B=1 TO 4
222 IF A+2*C(B)<11 THEN GO TO 230
225 IF (A(A+C(B))=C OR A(A+C(B))=H) AND A(A+2*C(B))=E
    THEN LET B=A+2*C(B): GO TO 280
230 PRINT AT 0,0; CHR$ (A(A+C(B)):
    IF A(A+C(B))=E THEN GO TO 260
240 NEXT B
250 GO TO 200
270 LET B=A+C(B)
280 PRINT AT 0,0;"EU EFECTUEZ MUTAREA DE LA ";
    CHR$ (64+INT(A/10));A-10*INT (A/10);" LA ";
    CHRS (64+INT (B/10)); B-10**INT (B/10);"
290 LET A(B)=C; LET A(A)=E
300 IF A(B+1)=H AND A (B+2)=C AND INT ((B+2)/10)=INT
    (B/10) THEN LET A (B+1)=E: LET CS=CS+1
310 IF A (B-1)=H AND A(B-2)=C AND INT
    ((B-2)/10)=INT (B/10) THEN LET A(B-1)=E:
    LET CS=CS+1
320 IF A>89 OR A<29 THEN RETURN
330 IF A(B+10)=H AND A(B+20)=C THEN LET A(B+10)=E:
    LET CS=CS+1
340 IF A(B-10)=H AND A(B-20)=C THEN LET A(B-10)=E:
    LET CS=CS+1
350 RETURN
370 BEEP 1,RND*20: PRINT AT 0,0;" ";AT 2,4;
    BRIGHT 1; INVERSE 1;"123456789"; INVERSE 0;
    AT 3,3;
375 FOR M=90 TO 10 STEP -10: PRINT TAB 3; INVERSE 1;
    CHR$ (M/10+64); INVERSE 0;
380 FOR N=1 TO 9
400 PRINT BRIGHT 1; INK M/30; CHR$ A(M+N);
410 NEXT N
420 PRINT INVERSE 1; CHR$ (M/10+64)
430 NEXT M: PRINT TAB 4; INVERSE 1;
    BRIGHT 1; "123456789"
440 PRINT TAB 3; "CALCULATORUL : "; CS
450 PRINT TAB 3; "JUCATORUL: "; H$
```

```
470 RETURN
480 IF CS>HS THEN PRINT "EU AM CISTIGAT": STOP
490 PRINT "DUMNEAVOASTRA ATI CISTIGAT"
500 STOP
520 INPUT "DUMNEAVOASTRA EFECTUATI MUTAREA DE LA
   (LITERA,CIFRA) ?";A$
525 IF A$="S" THEN STOP
530 INPUT "DE LA";(A$); "CATRE (LITERA,CIFRA)? ";B$
540 LET A=10*(CODE A$-64)+VAL A$ (2)
550 LET B=10*(CODE B$-64)+VAL B$ (2)
560 LET A(B)=H: LET A(A)=E
570 IF A(B+1)=C AND A(B+2)=H AND
    INT ((B+2)/10)=INT (B/10) THEN LET A(B+1)=E
    LET HS=HS+1
580 IF A(B-1)=C AND A(B-2)=H AND
    INT ((B-2)/10)=INT (B/10) THEN LET A(B-1)=E:
    LET HS=HS+1
590 IF B>79 THEN RETURN
600 IF A(B+10)=C AND A(B+20)=H THEN LET A (B+10)=E:
    LET HS=HS+1
605 IF B<31 THEN RETURN
610 IF A(B-10)=C AND A(B-20)=H THEN LET A(B-10)=E:
    LET HS=HS+1
620 RETURN
640 DIM A(129): DIM C(4): LET H=144: LET C=145:
    LET E=42
660 FOR Z=11 TO 29
670 IF Z=20 THEN LET Z=21
680 LET A(Z)=H
690 NEXT Z
700 FOR Z=31 TO 79
710 IF 10*INT (Z/10)=Z THEN LET Z=Z+1
720 LET A(Z)=E
730 NEXT Z
740 FOR Z=81 TO 99
750 IF Z=90 THEN LET Z=91
760 LET A(Z)=C
770 NEXT Z
780 LET HS=0
790 LET CS=0
810 FOR J=0 TO 1
820 LET M=144+J
830 FOR K=0 TO 7: READ N
840 POKE USR (CHR$ M)+K,N
850 NEXT K
860 NEXT J
870 FOR Z=1 TO 4
880 READ C(Z)
890 NEXT Z
900 DATA 56,124,126,126,126,124,56,0
```

```
910 DATA 0,126,126,126,126,126,126,0
920 DATA -10, -1,1,10
930 GO SUB 360
940 RETURN
```

15.10 Pirandello

Jocul "Pirandello " cunoscut şi sub numele de "Othello " este foarte asemănător jocului Reversi. Jocul se desfăşoară între doi parteneri, avînd ca suport o tablă de şah. In jocul Reversi, la momentul iniţial, fiecare dintre cei doi jucători îşi plasează cîte două piese în cele patru pătrate centrale. In jocul Pirandello, poziţiile primelor patru piese sînt fixate şi jocul începe cu aceste poziţii ocupate, aşa după cum veţi observa că apar pe ecran. Jocul se desfăsoară prin efectuarea de mutări alternativ de către cei doi parteneri conform cu următoarea regulă: o mutare constă în amplasarea unei noi piese, fiecare piesă nou amplasată trebuie să fie vecină cu cel puţin o piesă a adversarului iar la cealaltă extremitate a liniei să existe o piesă a jucătorului care efectuează mutarea. Linia de joc poate fi orizontală, verticală sau diagonală. Dacă piesa nou amplasată completează cel putin două linii de joc, atunci piesele din aceste linii se transformă în piese ale jucătorului care a efectuat mutarea. Scopul fiecărui jucător este de a determina cît mai multe transformări de piese adverse in piese proprii. Jocul se termină în momentul în care tabla de joc este completă sau cînd nu mai este posibilă o noua mutare; este declarat învingator jucătorul cu numărul maxim de piese pe tabla de joc. Pentru a indica o mutare, de exemplu în linia 4, coloana 3, se introduce numărul 43.

•P80

```
 10 REM ************************
 20 REM * P I R A N D E L L O *
 30 REM ************************
 40 GO TO 670
 50 LET y=145
 60 LET z=144
 70 LET k=0
 80 LET k=k+1
 90 LET b=e(k)
100 IF a(b) <> 146 THEN GO TO 290
110 LET j=0
120 FOR x=1 TO 8
130 LET n=d(x)
140 LET e=0
150 LET f=b
160 IF a(f+n) <> y THEN GO TO 200
170 LET e=1
180 LET f=f+n
190 GO TO 160
```

```
200 IF a(f+n) <> z THEN GO TO 260
210 IF e=0 THEN GO TO 260
220 FOR a=b TO f STEP n
230 LET a(a)=z
240 LET j=1
250 NEXT a
260 NEXT x
270 IF y=144 THEN GO TO 330
280 IF j=1 THEN GO TO 330
290 IF k <> 60 THEN GO TO 80
300 IF g=0 THEN GO TO 530
310 PRINT AT 0,0; FLASH 1; INK 2; PAPER 7;"Renunt la mutare."
320 BEEP 1,1: BEEP 1,5: BEEP 1,1: PRINT AT 0,0; TAB 25
330 PRINT AT 6,11; BRIGHT 1;"12345678"
340 BEEP .05,32: BEEP .05,12: BEEP .05,32
350 FOR a=1 TO 8
360 PRINT TAB 10; BRIGHT 1; INK 6;a;
370 FOR b=2 TO 9
380 LET w=a(a*10+b): BEEP .01,(w=144)+50*(w=145)-60*(w=146)
390 LET r=2*(w=144)+7*(w=145)+4*(w=146)
400 PRINT BRIGHT 1; INK r; CHR$ w;
410 NEXT b
420 PRINT BRIGHT 1; INK 6;a
430 NEXT a
440 PRINT TAB 11; INK 6; BRIGHT 1;"12345678"
450 IF y=144 THEN GO TO 50
460 LET y=144
470 LET z=145
480 INPUT INK 2; PAPER 6; FLASH 1; BRIGHT 1;"Indicati mutarea:";g
490 IF g=0 THEN GO TO 50
500 LET b=g+1
510 IF b<12 OR b>89 OR a(b) <> 146 THEN GO TO 480
520 GO TO 110
530 LET c=0
540 LET h=0
550 FOR a=1 TO 100
560 IF a(a)=144 THEN LET c=c+1
570 IF a(a)=145 THEN LET h=h+1
580 NEXT a
590 PRINT "Scorul final :","Eu  ";c,,"Dumneavoastra ";h
600 IF c>h THEN PRINT "Eu am cistigat ! Multumesc.": GO TO 630
610 IF c<h THEN PRINT "Dumneavoastra ati cistigat !": GO TO 630
620 IF c=h THEN PRINT "Remiza...Va multumesc pentru joc !"
630 INPUT "Apasati orice tasta pentru continuare";a$
640 INPUT "Doriti inca o partida ? (d/n) ";a$
650 IF a$="d" THEN CLS : RUN
660 STOP
670 BORDER 5: PAPER 2: INK 9: CLS
680 FOR a=1 TO 22: PRINT TAB a; INK a/5+3;"PIRANDELLO"
690 BEEP .01,1: BEEP .01,a*2: NEXT a
700 PRINT AT 5,5; FLASH 1; INK 7; PAPER 0;"Asteptati putin..."
710 PAPER 0: INK 9
720 DIM a(100): DIM e(60)
```

```
730 DIM d(8)
740 FOR a=1 TO 8
750 FOR b=2 TO 9
760 LET a(a*10+b)=146
770 NEXT b
780 NEXT a
790 FOR a=1 TO 60
800 READ b
810 LET e(a)=b
820 NEXT a
830 FOR a=1 TO 8
840 READ b
850 LET d(a)=b
860 NEXT a
870 LET a(45)=144: LET a(46)=145
880 LET a(56)=144: LET a(55)=145
890 LET y=144
900 READ a$: IF a$="Z" THEN BORDER 0: CLS : GO TO 330
910 FOR c=0 TO 7
920 READ b: POKE USR a$+c,b
930 NEXT c
940 GO TO 900
950 DATA 19,81,82,12,62,17,32,87,69,14,39,84,64,67,37,34;49,15
960 DATA 59,85,16,52,42,86,65,54,66,44,35,57,36,47
970 DATA 63,48,58,76,24,27,38,25,74,77,43,26,68,33,53
980 DATA 75,72,13,29,18,88,22,83,79,73,28,78,23
990 DATA 1,9,10,11,-1,-9,-10,-11
1000 DATA "A",0,24,60,126,126,60,24,0
1010 DATA "B",0,4,14,28,56,112,32,0
1020 DATA "C",255,129,129,129,129,129,129,255,"Z"
```

15.11 LABIRINT

Programul realizează definirea unui caracter utilizator şi deplasarea controlată a acestuia într-un labirint. Controlul este efectuat utilizînd tastele 5, 6, 7, 8.

• P81

```
   5 PRINT AT 10,10; FLASH 1;"PENTRU DEPLASAREA IN
     LABIRINT UTILIZATI TASTELE 5,6,7,8": PRINT:
     PRINT: FOR I=1 TO 10 STEP 2: BEEP I*RND, I*RND+I:
     NEXT I: PRINT "APASATI ORICE TASTA DACA DORITI
     CONTINUAREA": PAUSE 0: CLS
  10 GO SUB 8000
  20 GO SUB 7000
  30 PRINT INK RND*6".";
  40 GO SUB 6000
1000 REM Deplasarea "omuletului"
1020 PRINT AT edd,eda;" "
1025 IF SCREENS (dd,da)="." THEN LET scor=scor+2357
```

```
1030 PRINT AT dd,da; INK 2;a$
1032 IF ga=da AND gd=dd THEN GO TO 4000
1035 LET edd=dd: LET eda=da
1040 IF INKEY$="8" THEN IF da<13 THEN IF SCREEN$
     (dd,da+1)<>"X" THEN LET da=da+1: LET a$="B"
1050 IF INKEY$="5" THEN IF da>0 THEN IF SCREEN$
     (dd,da-1)<>"X" THEN LET da=da-1: LET a$="C"
1060 IF INKEY$="7" THEN IF dd>0 THEN IF SCREEN$
     (dd-1,da)<>"X" THEN LET dd=dd-1: LET a$="D"
1070 IF INKEY$="6" THEN IF dd<13 THEN IF SCREEN$
     (dd+1,da)<>"X" THEN LET dd=dd+1: LET a$="E"
1980 IF RND>.2 THEN GO TO 1000
1985 PRINT AT 7,16; FLASH 1; "SCORUL ESTE "; scor
2000 PRINT AT egd,ega; INK RND*6;"."
2010 PRINT AT gd,ga; INK 1;"G"
2012 IF ga=da AND gd=dd THEN GO TO 4000
2015 LET egd=gd: LET ega=ga
2020 IF dd<gd THEN IF SCREEN$ (gd-1,ga)<>"X" THEN LET
     gd=gd-1
2030 IF dd>gd THEN IF SCREEN$ (gd+1,ga)<>"X" THEN LET
     gd=gd+1
2040 IF da>ga THEN IF SCREEN$ (gd,ga+1)<>"X" THEN LET
     ga=ga+1
2050 IF da<ga THEN IF SCREEN$ (gd,ga-1)<>"X" THEN LET
     ga=ga-1
2999 GO TO 1000
4000 PRINT AT 17,0; INK RND*6; PAPER 9; FLASH 1;
     "SFIRSIT": FOR I=1 TO 10: BEEP RND*2, I*RND+2*I
4010 BEEP .01,RND*60: NEXT I
4020 PRINT INK RND*6; FLASH 1; AT egd,ega;" ";
     AT gd,ga;"█"
4030 PRINT INK RND*6; FLASH 1; AT edd,eda;" ";
     AT dd,da;"█"
4040 CLS: INPUT "DORITI RELUAREA (DA/NU) "; S$:
     IF S$="DA" THEN CLS: GO TO 10
4999 STOP
5000 REM DEFINIREA OMULETULUI
5010 FOR J=0 TO 7
5020 READ q
5030 POKE USR "B"+j,q
5040 NEXT j
5050 DATA BIN 00111100, BIN 01111111, BIN 11111100,
     BIN 11110000, BIN 11111000, BIN 11111100,
     BIN 01111111, BIN 00111100
5060 FOR j=0 TO 7
5070 READ q
5080 POKE USR "C"+j,q
5090 NEXT j
5100 DATA BIN 00111100,BIN 11111110,BIN 00111111,
     BIN 00011111, BIN 00001111,BIN 00111111,
```

```
     BIN 11111110, BIN 00111100
5110 FOR j=O TO 7
5120 READ q
5130 POKE USR "D"+j,q
5140 NEXT j
5150 DATA BIN 00111100, BIN 01111110, BIN 11111111,
     BIN 11111111,BIN 11110111,BIN 11100111,
     BIN 01000010, BIN 01000010
5160 FOR j=0 TO 7
5170 READ q
5180 POKE USR "E"+j,q
5190 NEXT j
5200 DATA BIN 01000010, BIN 01000010, BIN 11100111,
     BIN 11101111, BIN 11111111, BIN 11111111,
     BIN 01111110, BIN 00111100
5999 RETURN
6010 FOR j=0 TO 7
6020 READ q
6030 POKE USR "G"+j,q
6040 NEXT j
6050 DATA BIN 00111000, BIN 01111100, BIN 11010110,
     BIN 11010110, BIN 11111110, BIN 11111110,
     BIN 10101010, BIN 10101010
6999 RETURN
7010 FOR g=1 TO 14
7020 FOR h=1 TO 14
7030 PRINT INK RND*6;".";
7040 NEXT h
7050 PRINT
7060 NEXT g
7065 INVERSE 1
7070 PRINT AT 2,7;"XXXXX"
7080 PRINT AT 3,3;"X";AT 3,11;"X"
7090 PRINT AT 4,3;"X" AT 4,11;"X"
7100 PRINT AT 5,3;"XXXX";AT 5,9;"XXX"
7110 PRINT AT 6,6;"X";AT 6,9;"X"
7120 PRINT AT 7,6;"X";AT 7,9;"X"
7130 PRINT AT 8,3;"X";AT 8,6;"X";AT 8,9;"X"
7140 PRINT AT 10,3;"X"
7150 PRINT AT 11,3;"XXXX"
7160 PRINT AT 12,9;"XXX"
7165 INVERSE 0
7179 RETURN
8000 REM INITIALIZAREA VARIABILELOR
8010 LET DA=0
8020 LET DD=0
8030 LET EDA=0
8040 LET EDD=0
8050 LET GA=13
8060 LET GD=13
```

```
8070 LET EGA=13
8080 LET EGD=13
8090 LET SCOR=0
8100 LET AS="B"
8999 RETURN
```

16. DIVERSE

16.1 Perimetrul unei elipse

Plecînd de la ecuaţia unei elipse, exprimată în coordonate polare, [6] :

$x = a \cos \varphi$

$y = b \sin \varphi$

şi ştiind că lungimea unei curbe (dată de o ecuaţie în coordonate polare) este:

$$L = \int_{\varphi_1}^{\varphi_2} \sqrt{\left(\frac{dx}{d\varphi}\right)^2 + \left(\frac{d\varphi}{dx}\right)^2}\, d\varphi$$

, perimetrul unei elipse este dat de:

$$U = \int_0^{\frac{\pi}{2}} \sqrt{a^2 \sin^2\varphi + b^2 \cos^2\varphi}\, d\varphi = 4b \int_0^{\frac{\pi}{2}} \sqrt{1 - \frac{b^2 - a^2}{b^2} \sin^2\varphi}\, d\varphi$$

Integrala se aproximează folosind formula dreptunghiurilor pentru o diviziune cu N puncte a intervalului $[0, \frac{\pi}{2}]$.

• P82

```
  5 REM CALCULEAZA CU APROXIMATIE PERIMETRUL UNEI
     ELIPSE
 20 INPUT "VALOAREA PENTRU A = ";A
 21 INPUT "VALOAREA PENTRU B = ";B
 22 INPUT "VALOAREA PENTRU NUMARUL DE PUNCTE =";N
 23 CLS: PRINT AT 10,10; FLASH 1;
    "ASTEPTATI PUTIN PINA CALCULEZ !"
 30 LET P = PI
 40 LET H = P/2/N
 50 LET S = (1+A/B)/2
 60 FOR W = H TO P/2-H STEP H
 70 LET S = SQR (1-(B^2-A^2)* SIN(W)^2/B^2)+5
 80 NEXT W
 90 LET U = 4*B*S*H
 99 CLS
100 PRINT AT 10,10; FLASH 1; "VALOAREA APROXIMATIVA
     A PERIMETRULUI ESTE = "; FLASH 0;U
```

16.2 Metoda Simpson pentru calculul functiei ARCTG

Plecînd de la faptul că: $\operatorname{arctg} b = \int_0^b \frac{dx}{1 + x^2}$.

calculul integralei se face prin metoda Simpson în care numărul N de diviziuni a intervalului [0, b] este în mod necesar un mumăr par, [6]. Dacă $x_i = i \frac{b}{N}$ şi notăm

$y_i = = \frac{1}{1 + x^2}$, i=1,2,3, ..., N, atunci:

$$\int_0^b \frac{dx}{1+x^2} \approx \frac{b}{3N} \left[y_0 + y_N + 4 \left(y_1 + y_3 + \ldots + y_{N-1} \right) + 2 \left(y_2 + y_4 + \ldots y_{N-2} \right) \right]$$

•P83

```
  5 REM APROXIMATII PENTRU VALORILE FUNCTIEI ARCTG
      PRIN METODA SIMPSON
 20 INPUT "INTRODUCETI NUMARUL (IN MOD NECESAR PAR)
      DE PARTITII";N
 30 INPUT "ARGUMENTUL ESTE ";B
 40 LET H=B/N
 50 LET S=1+1/(1+B^2)
 60 FOR I=1 TO N-1 STEP 2
 70 LET X=I*H
 80 LET Y=1/(1+X^2)
 90 LET S=S+4*Y
100 NEXT I
110 FOR I=2 TO N-2 STEP 2
120 LET X=I*H
130 LET Y=1 /(1+H^2)
140 LET S=S+2*Y
150 NEXT I
160 PRINT AT 10,10; FLASH 1; "VALOARE APROXIMATIVA =";
      FLASH 0;S*H/3
165 FOR L=1 TO 30
170 BEEP .3*RND, 56*RND: NEXT L
180 INPUT "DORITI RELUAREA ?(DA/NU) ";A$:
      IF A$="DA" THEN CLS : GO TO 20
190 CLS: PRINT AT 10,10; FLASH 1; "GATA!"
200 FOR I=1 TO 20: BEEP O.3*RND,-10*RND: NEXT I:
      CLS: STOP
```

16.3 Media geometrică, aritmetică şi armonică a n numere

Fiind date n numere a_1, a_2, ..., a_n, avem:

$$\text{media aritmetică} = \frac{a_1 + a_2 + \dots + a_n}{n}$$

$$\text{media geometrică} = \left(a_1 \cdot a_2 \cdots a_n\right)^{\frac{1}{n}}$$

$$\text{media armonică} = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \dots + \frac{1}{a_n}}$$

•P84

```
 10 REM MEDIA GEOMETRICA, ARITMETICA, ARMONICA
    A N NUMERE
 30 INPUT "INDICATI CITE NUMERE SINT ",N
 40 DIM A (N)
 45 PRINT : PRINT FLASH 1; "INDICATI NUMERELE "
 50 FOR M=1 TO N
 60 INPUT ("A(";M;") = ");A(M)
 70 NEXT M
 80 CLS
 90 PRINT AT 4,4; FLASH 1;"MENIU"
100 PRINT AT 6,6; INK 2;"A PENTRU MEDIA ARITMETICA"
110 PRINT AT 8,6; INK 3; "G PENTRU MEDIA GEOMETRICA"
120 PRINT AT 10,6; INK 4;"R PENTRU MEDICA ARMONICA"
125 PRINT AT 12,6; "S PENTRU STOP"
140 LET A$ = INKEY$
150 IF INKEY$ ="A" THEN GO TO 200
160 IF INKEY$ ="G" THEN GO TO 300
170 IF INKEY$ ="R" THEN GO TO 400
175 IF INKEY$ ="S" THEN STOP
180 GO TO 140
200 CLS
210 LET S=0
220 FOR M=1 TO N
230 LET S=S+A(M)
240 NEXT M
250 LET S=S/N
260 PRINT AT 6,2;"MEDIA ARITMETICA ESTE ";S
270 PAUSE 100
275 CLS
280 GO TO 90
300 CLS
310 LET S=1
320 FOR M=1 TO N
```

```
330 LET S=S*A(M)
340 NEXT M
350 LET S=S ^(1/N)
360 PRINT AT 8,2;"MEDIA GEOMETRICA ESTE ";S
370 PAUSE 100
380 CLS
390 GO TO 90
400 CLS
410 LET S=0
420 FOR M=1 TO N
430 LET S=S+1/A(M)
440 NEXT M
450 LET S=N/S
460 PRINT AT 10,2;"MEDIA ARMONICA ESTE ";S
470 PAUSE 100
480 CLS
490 GO TO 90
```

16.4 Omotetie

Fiind dată o linie poligonală P_1, P_2, ..., P_n, transformarea omotetică a acesteia față de polul M cu raportul de omotetie r este linia poligonală T_1, T_2, ..., T_n obținută astfel încît $M \in P_iT_i$ și $\frac{MT_i}{MP_i} = r$, oricare ar fi i = 1, 2, ..., n .

●P85

```
  1 REM TRANSFORMA ORICE LINIE POLIGONALA PRIN
     OMOTETIE IN RAPORT CU UN POL DAT
 10 INPUT "INTRODUCETI ABSCISA RESPECTIV ORDONATA
     POLULUI",XA, YA
 20 PLOT XA,YA
 30 INPUT "INTRODUCETI RAPORTUL DE OMOTETIE",P
 40 INPUT "INTRODUCETI ABSCISA RESPECTIV ORDONATA
     PRIMULUI VIRF POLIGONAL",XB YB
 50 PLOT XB,YB
 60 INPUT "DIN CITE LINII FRINTE SE COMPUNE LINIA
     POLIGONALA",N
 70 DIM A(N): DIM B(N)
 80 FOR M=1 TO N
 90 INPUT "ORIZONTAL",A(M)
100 INPUT "VERTICAL",B(M)
110 DRAW A(M),B(M)
120 NEXT M
140 LET XM=XB-(XA-XB)*(P-1)
150 LET YM=YB-(YA-YB)*(P-1)
180 PLOT XM,YM
190 FOR M=1 TO N
```

```
200 DRAW P*A(M), P*B(M)
210 NEXT M
```

16.5 Analiza Fourier

O funcţie periodică, de perioadă T, continuă pe porţiuni, se poate pune sub forma:

$$y(x) = \frac{a_0}{2} + \sum_{j=1}^{\infty} a_j \cos j \frac{2\pi x}{T} + \sum_{j=1}^{\infty} b_j \sin j \frac{2\pi x}{T}$$

Dacă sînt cunoscute valorile funcţiilor $y_i = y(x_i)$, $x_i = i \frac{B - A}{n}$, se pune problema determinării coeficienţilor a_0, a_1, ... şi b_1, b_2, ... , care sînt date de formulele, [6]:

$$a_0 = \frac{2}{n} \sum_{i=1}^{n} y_i$$

$$a_j = \frac{2}{n} \sum_{i=1}^{n} y_i \cos j \frac{2\pi i}{n}$$

$$b_j = \frac{2}{n} \sum_{i=1}^{n} y_i \sin j \frac{2\pi i}{n}$$

●P86

```
  5 REM ANALIZA FOURIER
 20 INPUT "INTRODUCETI NUMARUL DE MASURATORI ",N
 30 LET P=PI
 40 INPUT "NUMARUL DE COEFICIENTI ESTE ";Z
 50 DIM Y (N)
 60 FOR I=1 TO N
 70 INPUT ("INTRODUCETI CEA DE A ";
    I;" - A VALOARE");Y(I)
 80 NEXT I
 81 PRINT AT 5,1;"S=1 DACA FUNCTIA ESTE PARA"
 82 PRINT AT 7,1; "S=2 DACA FUNCTIA ESTE IMPARA"
 83 PRINT AT 9,1;"ALTFEL S=3"
 90 INPUT "INDICATI VALOAREA PENTRU S=",S
 91 CLS
100 IF S=2 THEN GO TO 240
110 LET G=0
120 FOR I=1 TO N
130 LET G=G+Y(I)
140 NEXT I
```

```
145 BRIGHT 1
150 PRINT "AO= ";G*2/N
160 FOR J=1 TO Z
170 LET G=0
180 FOR I=1 TO N
190 LET G=G+Y(I)*COS (J*2*P*I/N)
200 NEXT I
210 PRINT "A";J;"= ";G*2/N
220 NEXT J
230 IF S=1 THEN STOP
240 FOR J=1 TO Z
250 LET G=0
260 FOR I=1 TO N
270 LET G=G+Y(I)*SIN(J*2*P*I/N)
280 NEXT I
290 PRINT "B";J;"=";G*2/N
300 NEXT J
```

16.6 Regresia liniară

Regresia liniară este una dintre tehnicile cel mai frecvent utilizate pentru ajustarea unor curbe pentru date rezultate dintr-un experiment.

Fie (X_i, Y_i), $1 \le i \le n$ datele rezultate în urma efectuării unui experiment prin care este observată variabila Y funcţie de variabila X. Acceptînd ca aproximaţie pentru tipul de dependenţă al lui Y de X dependenţa liniară, $Y = A + B*X$, parametrii A,B vor fi determinaţi astfel încît să se realizeze minimul expresiei

$$E = \sum_{i=1,n} (Y_i - (A+B*X_i))^2 .$$

Se poate demonstra că expresiile parametrilor A,B sînt date de :

$$A = \frac{\sum_{i=1,n} X_i^2 \sum_{i=1,n} Y_i - [\sum_{i=1,n} X_i \sum_{i=1,n} X_i Y_i]/n}{\sum_{i=1,n} X_i^2 - \sum_{i=1,n} X_i \sum_{j=1,n} X_j /n}$$

$$B = \frac{\sum_{i=1,n} X_i Y_i - \sum_{i=1,n} X_i \sum_{j=1,n} Y_j /n}{\sum_{i=1,n} X_i^2 - \sum_{i=1,n} X_i \sum_{j=1,n} X_j /n}$$

Calitatea ajustării este exprimată prin intermediul valorii coeficientului de corelație C dat de expresia :

$$C = \frac{n \sum_{i=1,n} X_i Y_i - \sum_{i=1,n} X_i \sum_{i=1,n} Y_i}{\left\{ \left[n \sum_{i=1,n} X_i^2 - \left(\sum_{i=1,n} X_i \right)^2 \right] \left[n \sum_{i=1,n} Y_i^2 - \left(\sum_{i=1,n} Y_i \right)^2 \right] \right\}^{\frac{1}{2}}}$$

• P87

```
1002 REM ******************************************
1005 PRINT "REGRESIA LINEARA SIMPLA "                *
1016 REM ******************************************
1017 REM * Programul determina coeficientii dreptei   *
1018 REM * de regresie Y=A+B*X,unde X este variabi-   *
1019 REM * la independenta,Y variabila dependenta.    *
1020 REM * In cazul in care numarul datelor de ob-    *
1021 REM * servatie este mai mare de 30,este nece-    *
1022 REM * sara modificarea instructiunilor DIM.      *
1023 REM * Numarul de observatii este N.Datele de     *
1024 REM * observatie sint memorate in X,Y            *
1025 REM ******************************************
1030 DIM x(30)
1035 DIM y(30)
1040 PRINT: INPUT "Numarul de observatii n=  ";n
1045 IF n>3 THEN GOTO 1070
1050 PRINT "EROARE ! Valoarea lui n>3 ":PRINT
1055 INPUT "Doriti reluarea ?(d/n)";a$
1060 IF a$="d" THEN GOTO 1040
1065 STOP
1070 PRINT "Introduceti valorile perechilor X,Y "
1075 FOR i=1 TO n
1080 PRINT "Punctul ";i
1085 INPUT "X = ";x(i)
1090 INPUT "Y = ";y(i)
1095 NEXT i
1100 PRINT:PRINT "Optiunile posibile"
1110 PRINT TAB(7); "1-Listarea datelor introduse"
1115 PRINT TAB(7); "2-Modificarea datelor "
1120 PRINT TAB(7); "3-Regresia"
1125 PRINT TAB(7); "4-Terminat"
1130 INPUT "Optiunea : ";op :PRINT
1135 IF (op<1)OR (op>4) THEN PRINT "Repetati!":GOTO 1100
1140 IF op=1 THEN GOSUB 1330
```

```
1145 IF op=2 THEN GOSUB 1450
1150 IF op=3 THEN GOSUB 1520
1155 IF op=4 THEN GOTO 1870
1160 GOTO 1100
1300 REM ****************************************
1310 REM* Subprogram pentru listarea datelor introduse*
1320 REM ****************************************
1330 PRINT: PRINT " Datele introduse "
1340 PRINT "   X","   Y"
1350 LET ic=1
1360 FOR i=1 TO n
1370 IF i<>(ic*15) THEN GOTO 1400
1375 LET ic=ic+1
1380 PRINT: INPUT "Apasati orice tasta pentru
     continuare";a$
1390 PRINT
1400 PRINT x(i),y(i)
1410 NEXT i
1420 RETURN
1430 REM ****************************************
1435 REM * Subprogram pentru modificarea datelor     *
1440 REM ****************************************
1450 PRINT: INPUT " Indicele punctului de modificat ";d
1460 PRINT "Noile valori pentru punctul ";d
1470 INPUT "X =";x(d):INPUT "Y =";y(d)
1480 INPUT "Mai sint valori de modificat ?(d/n)";a$
1490 IF a$="d" THEN GOTO 1450
1500 RETURN
1510 REM ****************************************
1515 REM * Subprogram regresia lineara               *
1520 REM ****************************************
1530 LET sx=0: LET sy=0: LET sxy=0: LET sx2=0: LET sy2=0
1540 FOR i=1 TO n
1550 LET sx=sx+x(i): LET sy=sy+y(i)
1560 LET sx2=sx2+x(i)*x(i): LET sy2=sy2+y(i)*y(i)
1570 LET sxy=sxy+x(i)*y(i)
1600 NEXT i
1610 LET b=(n*sxy-sx*sy)/(n*sx2-sx*sx)
1620 LET a=(sy-b*sx)/n
1630 REM Calculul coeficientului de corelatie
1640 LET c=(sxy-sx*sy/n)/(SQR((sx2-sx*sx/n)*
     (sy2-sy*sy/n)))
1650 LET cr=c*c
1660 LET sse=sy2-sy*sy/n- b*(sxy-sx*sy/n)
1670 LET se=SQR(sse/(n-2))
1680 REM ****************************************
1685 REM * Subprogram pentru afisarea rezultatelor   *
1690 REM ****************************************
1700 PRINT :PRINT "Ecuatia dreptei de regresie :"
1710 PRINT "Y =";a;"+";b;"*X" :PRINT
```

```
1720 PRINT " Coeficientul de corelatie = ";c
1730 PRINT "Coeficientul de determinare = ";cr
1740 PRINT " Ecartul =  ";se
1750 PRINT PRINT "Valorile reale si valorile estimate :"
1760 PRINT "X","Y","Y estimat ","Eroarea"
1770 LET ic=1
1780 FOR i=1 TO n
1790 IF i<>(ic*15) THEN GOTO 1820
1800 PRINT:INPUT "Apasati orice tasta pentru
     continuare";a$
1810 PRINT: LET ic=ic+1
1820 LET ey=a+b*x(i)
1830 PRINT x(i),y(i),ey,y(i)-ey
1840 NEXT i
1850 PRINT:
     INPUT "Apasati orice tasta pentru continuare";a$
1860 PRINT:RETURN
1870 PRINT:PRINT TAB(7); "Terminat"
```

ANEXA 1

Codurile ASCII corespunzătoare setului de caractere BASIC

Cod	Caracter	Cod	Caracter
0	neutilizat	32	spaţiu liber (blank)
1	neutilizat	33	!
2	neutilizat	34	"
3	neutilizat	35	#
4	neutilizat	36	$
5	neutilizat	37	%
6	PRINT	38	&
7	EDIT	39	'
8	cursor stînga	40	(
9	cursor dreapta	41	)
10	cursor jos	42	*
11	cursor sus	43	+
12	DELETE	44	,
13	ENTER	45	-
14	număr	46	.
15	neutilizat	47	/
16	INK control	48	0
17	PAPER control	49	1
18	FLASH control	50	2
19	BRIGHT control	51	3
20	INVERSE control	52	4
21	OVER control	53	5
22	AT control	54	6
23	TAB control	55	7
24	neutilizat	56	8
25	neutilizat	57	9
26	neutilizat	58	:
27	neutilizat	59	;
28	neutilizat	60	<
29	neutilizat	61	=
30	neutilizat	62	>
31	neutilizat	63	?

Cod	Caracter	Cod	Caracter	Cod	Caracter
64	@	93	]	122	z
65	A	94	^	123	{
66	B	95	_	124	|
67	C	96	£	125	}
68	D	97	a	126	–
69	E	98	b	127	©
70	F	99	c	128	
71	G	100	d	129	▝
72	H	101	e	130	▘
73	I	102	f	131	▀
74	J	103	g	132	▗
75	K	104	h	133	▐
76	L	105	i	134	▚
77	M	106	j	135	▜
78	N	107	k	136	▖
79	O	108	l	137	▞
80	P	109	m	138	▌
81	Q	110	n	139	▛
82	R	111	o	140	▄
83	S	112	p	141	▟
84	T	113	q	142	▙
85	U	114	r	143	█
86	V	115	s	pentru caractere grafice definite de utilizator: 144 - 164	
87	W	116	t		
88	X	117	u	144	(a)
89	Y	118	v	145	(b)
90	Z	119	w	146	(c)
91	[	120	x	147	(d)
92	/	121	y	148	(e)

Cod	Caracter	Cod	Caracter
149	(f)	174	VAL$
150	(g)	175	CODE
151	(h)	176	VAL
152	(i)	177	LEN
153	(j)	178	SIN
154	(k)	179	COS
155	(l)	180	TAN
156	(m)	181	ASN
157	(n)	182	ACS
158	(o)	183	ATN
159	(p)	184	LN
160	(q)	185	EXP
161	(r)	186	INT
162	(s)	187	SQR
163	(t)	188	SGN
164	(u)	189	ABS
		190	PEEK
165	RND	191	INK
166	INKEY$	192	USR
167	PI	193	STR$
168	FN	194	CHR$
169	POINT	195	NOT
170	SCREEN$	196	BIN
171	ATTR	197	OR
172	AT	198	AND
173	TAB	199	< =

Cod	Caracter	Cod	Caracter
200	> =	228	DATA
201	< >	229	RESTORE
202	LINE	230	NEW
203	THEN	231	BORDER
204	TO	232	CONTINUE
205	STEP	233	DIM
206	DEF FN	234	REM
207	CAT	235	FOR
208	FORMAT	236	GO TO
209	MOVE	237	GO SUB
210	ERASE	238	INPUT
211	OPEN#	239	LOAD
212	CLOSE#	240	LIST
213	MERGE	241	LET
214	VERIFY	242	PAUSE
215	BEEP	243	NEXT
216	CIRCLE	244	POKE
217	INK	245	PRINT
218	PAPER	246	PLOT
219	FLASH	247	RUN
220	BRIGHT	248	SAVE
221	INVERSE	249	RANDOMIZE
222	OVER	250	IF
223	OUT	251	CLS
224	LPRINT	252	DRAW
225	LLIST	253	CLEAR
226	STOP	254	RETURN
227	READ	255	COPY

ANEXA 2
Lista programelor

BIBLIOGRAFIE

[1] Apps Vince, **40 Educational Games for the SPECTRUM**, Granada Publishing, 1983.

[2] Bucur C. M., **Metode numerice**, Editura Facla, 1983.

[3] Busch Rudolf, **BASIC für Einsteiger**, Franzis-Verlag, 1984.

[4] Crouzeix M., Mignot A. L., **Analyse numérique des écuations différentielles**, Masson, 1984.

[5] Crouzeix M., Mignot A. L., **Exercices d'analyse numérique des écuations différentielles**, Masson, 1986.

[6] Demmig Gudrun, **Programmieren-Leicht Gemacht**, vol.1-2, Demmig Verlag KG, 1977, 1978.

[7] Dumitrașcu Liviu, **Învățăm microelectronică interactivă**, vol.1-2, Editura Tehnică, București, 1989.

[8] Hartnell Tim, **Dynamic Games for the ZX SPECTRUM**, Sinclair Browne Ltd., 1983.

[9] Hartnell Tim, **Getting Started on Your SPECTRUM**, Futura, Macdonald & Co, 1983.

[10] Hartnell Tim, **La conduite du ZX-SPECTRUM, MICRO-ORDINATEURS**, 1984.

[11] Jamshidi M., Malek-Zavarei M., **Linear Control Systems**, Pergamon Press, 1986.

[12] Lien A. David, **The BASIC Handbook**, Compusoft Publ., 1981.

[13] Marinescu Gh., Rizzoli I., Popescu I., Ștefan C., **Probleme de analiză numerică rezolvate cu calculatorul**, Editura Academiei, 1987.

[14] Petrescu A. et. al., **Totul despre calculatorul personal a-MIC**, vol.1-2, Editura Tehnică, 1985.

[15] Renko Hal, Edwards Sam, **Spectacular Games for your ZX SPECTRUM**, Addison Wesley, Publ. Co., 1983.

[16] Rosenfelder Lewis, **BASIC. Faster & Better & Other Mysteries**, IJG Inc., 1981.

[17] Schmidt Klauss, Stickler Wolfgang, **Programmieren in BASIC. Vom Problem zum Algorithmus**, Verlag Harri Deutch, 1985.

[18] Schupp Wilfred, **Schüler Programmieren in BASIC**, Ferdinand Schoningh at Paderborn, 1985.

[19] Shampine L.F., Allen R.C., **Numerical Computing. An Introduction,** Saunders, Philadelphia, 1973.

[20] Shampine L. F., Gordon M. K., **Computer Solution of Ordinary Differential quations. The Initial Value Problem,** W.H. Freeman & Co., San Francisco, 1975.

[21] Torning W., **Numerische Mathematik für Ingenieure und Physiker**, Springer Verlag, 1979.

[22] Vickers Steven, **ZX-SPECTRUM BASIC Programming**; Sinclair Research, 1982.

[23] Wolfe Philip, Koelling, **Programmes BASIC pour ingénieurs et scientifiques**, Prentice Hall, International, 1986.

[24] *** **Introduction to BASIC,** Digital Equipment Corporation Maynard, Massachusetts, 1978.

[25] *** **ZX-SPECTRUM + USER GUIDE**, Sinclair Research Ltd. Dorling Kindersley, 1986.

[26] *** **TIM-S - Manual de funcţionare şi utilizare**, I.T.C.I. - F.M.E.C.T.C.

Tiparul executat sub comanda nr. 20.515
Regia Autonomă a Imprimeriilor
Imprimeria CORESI
Piaţa Presei Libere, 1, Bucureşti
ROMÂNIA

ISBN 973-95626-6-3

Hello, BASIC

este o excelentă introducere în lumea programării limbajului BASIC, limbaj recomandat începătorilor. Prin limbajul BASIC lucrarea ne introduce şi în elemente de grafică, animaţie şi efecte sonore pe calculator. Cele 85 de programe, avînd 4 grade de dificultate, fac parte dintre numeroasele alte exemple din carte. Programele sînt rulate pe calculatoare compatibile **SINCLAIR-SPECTRUM (HC, CIP, JET, COBRA, TIM)**, dar ele pot fi uşor adaptate pentru alte dialecte BASIC pe alte tipuri de calculatoare (**PC**).

Despre autor:

Doamna Luminiţa State este conferenţiară la Catedra de Informatică a Facultăţii de Matematică, Universitatea Bucureşti. Predă din 1977 cursuri de inteligenţă artificială, recunoaşterea formelor, laboratoare şi seminarii de BASIC, LISP, PROLOG şi fundamentele informaticii.

Preţ: 550 lei